

1 [JScript] function PreAuthenticate(request : WebRequest, credentials :  
2 ICredentials) : Authorization;  
3  
4

4 *Description*

5        Returns an instance of the **System.Net.Authorization** class for an  
6 authentication request to a server.

7 *Return Value:* An **System.Net.Authorization** instance containing the  
8 authorization message for the request.

9        When the **System.Net.IAuthenticationModule.CanPreAuthenticate**  
10 property is **true** , the  
11 **System.Net.IAuthenticationModule.PreAuthenticate(System.Net.WebRequest**  
12 **,System.Net.ICredentials)** method will return an instance of the  
13 **System.Net.Authorization** class containing an authentication message. The  
14 **System.Net.WebRequest** instance associated with the authentication request. The  
15 credentials associated with the authentication request.

16        **ICertificatePolicy** interface (System.Net)

17        **PreAuthenticate**

18  
19  
20 *Description*

21        Validates a server certificate.

22        The **System.Net.ICertificatePolicy** interface is used to provide custom  
23 security certificate validation for an application. The default policy is to allow  
24 valid certificates, as well as valid certificates that have expired. To change this  
25 policy, implement the **System.Net.ICertificatePolicy** interface with a different

1 policy, and then assign that policy to  
2 **System.Net.ServicePointManager.CertificatePolicy** .  
3     CheckValidationResult  
4  
5 [C#] bool CheckValidationResult(ServicePoint srvPoint, X509Certificate  
6 certificate, WebRequest request, int certificateProblem);  
7 [C++] bool CheckValidationResult(ServicePoint\* srvPoint, X509Certificate\*  
8 certificate, WebRequest\* request, int certificateProblem);  
9 [VB] Function CheckValidationResult(ByVal srvPoint As ServicePoint, ByVal  
10 certificate As X509Certificate, ByVal request As WebRequest, ByVal  
11 certificateProblem As Integer) As Boolean  
12 [JScript] function CheckValidationResult(srvPoint : ServicePoint, certificate :  
13 X509Certificate, request : WebRequest, certificateProblem : int) : Boolean;  
14

15 *Description*

16     Validates a server certificate.

17 *Return Value*: **true** if the certificate should be honored; otherwise, **false** .

18     The

19 **System.Net.ICertificatePolicy.CheckValidationResult(System.Net.ServicePoi**  
20 **nt, System.Security.Cryptography.X509Certificates.X509Certificate, System.N**  
21 **et.WebRequest, System.Int32)** method implements the application certificate  
22 validation policy. The method can examine the *srvPoint* , *certificate* , *request* , and  
23 *certificateProblem* parameters to determine whether the certificate should be  
24 honored. The **System.Net.ServicePoint** that will use the certificate. The certificate

1 to validate. The request that received the certificate. The problem encountered  
2 when using the certificate.

3 ICredentials interface (System.Net)

4 CheckValidationResult

5

6

7 *Description*

8 Provides the base authentication interface for retrieving credentials for Web  
9 client authentication.

10 The **System.Net.ICredentials** interface provides the  
11 **System.Net.ICredentials.GetCredential(System.Uri, System.String)** method to  
12 objects that supply network credentials to applications.

13 GetCredential

14

15 [C#] NetworkCredential GetCredential(Uri uri, string authType);  
16 [C++] NetworkCredential\* GetCredential(Uri\* uri, String\* authType);  
17 [VB] Function GetCredential( ByVal uri As Uri, ByVal authType As String) As  
18 NetworkCredential  
19 [JScript] function GetCredential(uri : Uri, authType : String) : NetworkCredential;

20

21 *Description*

22 Returns a **System.Net.NetworkCredential** object that is associated with  
23 the specified URI, and authentication type.

24 *Return Value:* The **System.Net.NetworkCredential** associated with the specified  
25 URI and authentication type, or if no credentials are available, **null** .

1        The **System.Net.ICredentials.GetCredential(System.Uri, System.String)**  
2        method returns a **System.Net.NetworkCredential** instance that contains the  
3        credentials associated with the specified URI and authorization scheme. When no  
4        credentials are available, the  
5        **System.Net.ICredentials.GetCredential(System.Uri, System.String)** method  
6        returns **null**. The **System.Uri** that the client is providing authentication for. The  
7        type of authentication, as defined in the  
8        **System.Net.IAuthenticationModule.AuthenticationType** property.

9            IPAddress class (System.Net)

10          GetCredential

13        *Description*

14        Provides an Internet Protocol (IP) address.

15        The **System.Net.IPEndPoint** class contains the address of a computer on an  
16        IP network.

17          GetCredential

19        [C#] public static readonly IPAddress Any;

20        [C++] public: static IPAddress\* Any;

21        [VB] Public Shared ReadOnly Any As IPAddress

22        [JScript] public static var Any : IPAddress;

24        *Description*

1 Provides an IP address indicating that the server should listen for client  
2 activity on all network interfaces. This field is read-only.

3 The **System.Net.Sockets.Socket.Bind(System.NetEndPoint)** method  
4 uses the **System.Net.IPEndPoint.Any** field to indicate that a  
5 **System.Net.Sockets.Socket** instance should listen for client activity on all  
6 network interfaces.

7 **GetCredential**

8  
9 [C#] public static readonly IPAddress Broadcast;  
10 [C++] public: static IPAddress\* Broadcast;  
11 [VB] Public Shared ReadOnly Broadcast As IPAddress  
12 [JScript] public static var Broadcast : IPAddress;

13  
14 *Description*

15 Provides the IP broadcast address. This field is read-only.

16 The **System.Net.IPEndPoint.Broadcast** field is equivalent to  
17 255.255.255.255.

18 **GetCredential**

19  
20 [C#] public static readonly IPAddress Loopback;  
21 [C++] public: static IPAddress\* Loopback;  
22 [VB] Public Shared ReadOnly Loopback As IPAddress  
23 [JScript] public static var Loopback : IPAddress;

24  
25 *Description*

1 Provides the IP loopback address. This field is read-only.  
2 The **System.Net.IPEndPoint.Loopback** field is equivalent to 127.0.0.1 in  
3 network byte order.

4 **GetCredential**

5  
6 [C#] public static readonly IPAddress None;  
7 [C++] public: static IPAddress\* None;  
8 [VB] Public Shared ReadOnly None As IPAddress  
9 [JScript] public static var None : IPAddress;

10  
11 *Description*

12 Provides an IP address indicating that no network interface should be used.  
13 This field is read-only.  
14 The **System.Net.Sockets.Socket.Bind(System.Net.EndPoint)** method  
15 uses the **System.Net.IPEndPoint.None** field to indicate that a  
16 **System.Net.Sockets.Socket** instance should not listen for client activity.

17 **IPAddress**

18 *Example Syntax:*

19 **GetCredential**

20  
21 [C#] public IPAddress(long newAddress);  
22 [C++] public: IPAddress(\_int64 newAddress);  
23 [VB] Public Sub New(ByVal newAddress As Long)  
24 [JScript] public function IPAddress(newAddress : long);  
25

1      *Description*

2          Initializes a new instance of the **System.Net.IPAddress** class with the  
3          specified address.

4          The **System.Net.IPAddress** instance is created with the  
5          **System.Net.IPAddress.Address** property set to *newaddress* . The integer value of  
6          the IP address.

7          Address

8          GetCredential

9          

10     [C#] public int Address {get; set;}

11     [C++] public: \_\_property int get\_Address();public: \_\_property void  
12     set\_Address(int);

13     [VB] Public Property Address As Integer

14     [JScript] public function get Address() : int;public function set Address(int);

15      *Description*

16          An Internet Protocol (IP) address.

17          To convert **System.Net.IPAddress.Address** to dotted-quad notation, use  
18          the **System.Net.IPAddress.ToString** method.

19          AddressFamily

20          GetCredential

21     [C#] public AddressFamily AddressFamily {get;}

22     [C++] public: \_\_property AddressFamily get\_AddressFamily();

1 [VB] Public ReadOnly Property AddressFamily As AddressFamily

2 [JScript] public function get AddressFamily() : AddressFamily;

3

4 *Description*

5 Specifies the address family of the IP address.

6 Equals

7

8 [C#] public override bool Equals(object comparand);

9 [C++] public: bool Equals(Object\* comparand);

10 [VB] Overrides Public Function Equals(ByVal comparand As Object) As Boolean

11 [JScript] public override function Equals(comparand : Object) : Boolean;

12

13 *Description*

14 Compares two IP addresses.

15 *Return Value:* **true** if the two address are equal; otherwise, **false** .

16 The **System.Net.IPEndPoint.Equals(System.Object)** method compares the  
17 current **System.Net.IPEndPoint** instance with the *comparand* parameter and  
18 returns **true** if the two instance contain the same IP address. An  
19 **System.Net.IPEndPoint** instance to compare to the current instance.

20 GetHashCode

21

22 [C#] public override int GetHashCode();

23 [C++] public: int GetHashCode();

24 [VB] Overrides Public Function GetHashCode() As Integer

25 [JScript] public override function GetHashCode() : int;

1        *Description*

2                Returns a hash value for an IP address.

3        *Return Value:* An integer hash value.

4        The **System.Net.IPEndPoint.GetHashCode** method returns a hash code of  
5        the IP address. This value can be used as a key in hash tables.

6        HostToNetworkOrder

7        [C#] public static short HostToNetworkOrder(short host);

8        [C++] public: static short HostToNetworkOrder(short host);

9        [VB] Public Shared Function HostToNetworkOrder( ByVal host As Short) As  
10                Short

11        [JScript] public static function HostToNetworkOrder(host : Int16) : Int16;

12        *Description*

13                Converts a short value from host byte order to network byte order.

14        *Return Value:* A short value expressed in network byte order.

15                Different computers use different conventions for ordering the bytes within  
16                multi-byte integer values. Some computers put the most significant byte first  
17                (known as big-endian order) and others put the least-significant byte first (known  
18                as little-endian order). To enable computers that use different byte ordering, all  
19                integer values sent over the network are sent in network byte order. The number to  
20                convert expressed in host byte order.

21        HostToNetworkOrder

```
1
2 [C#] public static int HostToNetworkOrder(int host);
3 [C++] public: static int HostToNetworkOrder(int host);
4 [VB] Public Shared Function HostToNetworkOrder( ByVal host As Integer) As
5 Integer
6 [JScript] public static function HostToNetworkOrder(host : int) : int;
```

```
7
8 Description
```

9 Converts an integer value from host byte order to network byte order.

10 *Return Value:* An integer value expressed in network byte order.

11 Different computers use different conventions for ordering the bytes within  
12 multi-byte integer values. Some computers put the most significant byte first  
13 (known as big-endian order) and others put the least-significant byte first (known  
14 as little-endian order). To enable computers that use different byte ordering, all  
15 integer values sent over the network are sent in network byte order. The number to  
16 convert expressed in host byte order.

```
17 HostToNetworkOrder
```

```
18
19 [C#] public static long HostToNetworkOrder(long host);
20 [C++] public: static __int64 HostToNetworkOrder( __int64 host);
21 [VB] Public Shared Function HostToNetworkOrder( ByVal host As Long) As
22 Long
23 [JScript] public static function HostToNetworkOrder(host : long) : long; Converts
24 a value from host byte order to network byte order.
```

1  
2 *Description*

3 Converts a long value from host byte order to network byte order.

4 *Return Value:* A long value expressed in network byte order.

5 Different computers use different conventions for ordering the bytes within  
6 multi-byte integer values. Some computers put the most significant byte first  
7 (known as big-endian order) and others put the least-significant byte first (known  
8 as little-endian order). To enable computers that use different byte ordering, all  
9 integer values sent over the network are sent in network byte order. The number to  
10 convert expressed in host byte order.

11 IsLoopback

12  
13 [C#] public static bool IsLoopback(IPAddress address);

14 [C++] public: static bool IsLoopback(IPAddress\* address);

15 [VB] Public Shared Function IsLoopback( ByVal address As IPAddress) As

16 Boolean

17 [JScript] public static function IsLoopback(address : IPAddress) : Boolean;

18  
19 *Description*

20 Indicates whether the specified IP address is the loopback address.

21 *Return Value:* **true** if *address* is the loopback address; otherwise **false** .

22 The **System.Net.IPEndPoint.IsLoopback(System.Net.IPEndPoint)** method  
23 compares *address* to **System.Net.IPEndPoint.Loopback** and returns **true** if the  
24 two IP address are the same. An IP address.

25 NetworkToHostOrder

```
1
2 [C#] public static short NetworkToHostOrder(short network);
3 [C++] public: static short NetworkToHostOrder(short network);
4 [VB] Public Shared Function NetworkToHostOrder(ByVal network As Short) As
5 Short
6 [JScript] public static function NetworkToHostOrder(network : Int16) : Int16;
```

```
7
8 Description
```

9 Converts a short value from network byte order to host byte order.

10 *Return Value:* A short value expressed in host byte order.

11 Different computers use different conventions for ordering the bytes within  
12 multi-byte integer values. Some computers put the most significant byte first  
13 (known as big-endian order) and others put the least-significant byte first (known  
14 as little-endian order). To enable computers that use different byte ordering, all  
15 integer values sent over the network are sent in network byte order. The number to  
16 convert expressed in network byte order.

```
17 NetworkToHostOrder
```

```
18
19 [C#] public static int NetworkToHostOrder(int network);
20 [C++] public: static int NetworkToHostOrder(int network);
21 [VB] Public Shared Function NetworkToHostOrder(ByVal network As Integer)
22 As Integer
23 [JScript] public static function NetworkToHostOrder(network : int) : int;
```

```
24
25 Description
```

1 Converts an integer value from network byte order to host byte order.

2 *Return Value:* An integer value expressed in host byte order.

3 Different computers use different conventions for ordering the bytes within  
4 multi-byte integer values. Some computers put the most significant byte first  
5 (known as big-endian order) and others put the least-significant byte first (known  
6 as little-endian order). To enable computers that use different byte ordering, all  
7 integer values sent over the network are sent in network byte order. The number to  
8 convert expressed in network byte order.

9 NetworkToHostOrder

10  
11 [C#] public static long NetworkToHostOrder(long network);  
12 [C++] public: static \_\_int64 NetworkToHostOrder(\_\_int64 network);  
13 [VB] Public Shared Function NetworkToHostOrder( ByVal network As Long ) As  
14 Long  
15 [JScript] public static function NetworkToHostOrder( network : long ) : long;

16 Converts a number from network byte order to host byte order.

17  
18 *Description*

19 Converts a long value from network byte order to host byte order.

20 *Return Value:* A long value expressed in host byte order.

21 Different computers use different conventions for ordering the bytes within  
22 multi-byte integer values. Some computers put the most significant byte first  
23 (known as big-endian order) and others put the least-significant byte first (known  
24 as little-endian order). To enable computers that use different byte ordering, all

1 integer values sent over the network are sent in network byte order. The number to  
2 convert expressed in network byte order.

3 **Parse**

4

5 [C#] public static IPAddress Parse(string ipString);  
6 [C++] public: static IPAddress\* Parse(String\* ipString);  
7 [VB] Public Shared Function Parse(ByVal ipString As String) As IPAddress  
8 [JScript] public static function Parse(ipString : String) : IPAddress;

9

10 *Description*

11 Converts an IP address string to an **System.Net IPAddress** instance.

12 *Return Value:* An **System.Net IPAddress** instance.

13 The static **System.Net IPAddress.Parse(System.String)** method creates  
14 an **System.Net IPAddress** instance from an IP address expressed in dotted-quad  
15 ("192.168.1.2") notation. A string containing an IP address in dotted-quad notation  
16 (for example, "192.168.1.2").

17 **ToString**

18

19 [C#] public override string ToString();  
20 [C++] public: String\* ToString();  
21 [VB] Overrides Public Function ToString() As String  
22 [JScript] public override function ToString() : String;

23

24 *Description*

1 Converts an Internet address to standard dotted-quad format.  
2 *Return Value:* A string containing the IP address in dotted-quad format (for  
3 example, "192.168.1.2").

4 The **System.Net.IPEndPoint.ToString** method converts the IP address  
5 stored in the **System.Net.IPEndPoint.Address** property is converted to dotted-  
6 quad notation (for example, "192.168.1.2").

7 **IPEndPoint** class (System.Net)

8 **ToString**

9  
10  
11 *Description*

12 Represents a network endpoint as an IP address and a port number.

13 The **System.Net.IPEndPoint** class contains the host and port information  
14 needed by an application to connect to a service on a host. By combining the host's  
15 IP address and port number of a service, the **System.Net.IPEndPoint** class forms  
16 a connection point to a service.

17 **ToString**

18  
19 [C#] public const int MaxPort;  
20 [C++] public: const int MaxPort;  
21 [VB] Public Const MaxPort As Integer  
22 [JScript] public var MaxPort : int;

23  
24 *Description*  
25

1        Specifies the maximum value that can be assigned to the  
2 **System.Net.IPEndPoint.Port** property. This field is read-only.

3        **ToString**

5        [C#] public const int MinPort;

6        [C++] public: const int MinPort;

7        [VB] Public Const MinPort As Integer

8        [JScript] public var MinPort : int;

10      *Description*

11      Specifies the minimum value that can be assigned to the  
12 **System.Net.IPEndPoint.Port** property. This field is read-only.

13      **IPEndPoint**

14      *Example Syntax:*

15      **ToString**

17      [C#] public IPEndPoint(long address, int port);

18      [C++] public: IPEndPoint(\_\_int64 address, int port);

19      [VB] Public Sub New(ByVal address As Long, ByVal port As Integer)

20      [JScript] public function IPEndPoint(address : long, port : int); Initializes a new  
21      instance of the **System.Net.IPEndPoint** class.

23      *Description*

1       Initializes a new instance of the **System.Net.IPEndPoint** class with the  
2       specified address and port number. The IP address of the Internet host. The port  
3       number associated with the address, or 0 to specify any available port.

4       **IPEndPoint**

5       *Example Syntax:*

6       **ToString**

7

8       [C#] public IPEndPoint(IPAddress address, int port);

9       [C++] public: IPEndPoint(IPAddress\* address, int port);

10      [VB] Public Sub New(ByVal address As IPAddress, ByVal port As Integer)

11      [JScript] public function IPEndPoint(address : IPAddress, port : int);

12

13      *Description*

14       Initializes a new instance of the **System.Net.IPEndPoint** class with the  
15       specified address and port number. The IP address of the Internet host. The port  
16       number associated with *address* .

17       **Address**

18       **ToString**

19

20      [C#] public IPAddress Address {get; set;}

21      [C++] public: \_\_property IPAddress\* get\_Address(); public: \_\_property void  
22       set\_Address(IPAddress\*);

23      [VB] Public Property Address As IPAddress

24      [JScript] public function get Address() : IPAddress; public function set  
25       Address(IPAddress);

1  
2 *Description*  
3     Gets or sets the IP address of the end point.  
4     AddressFamily  
5     ToString  
6  
7 [C#] public override AddressFamily AddressFamily {get;}  
8 [C++] public: \_\_property virtual AddressFamily get\_AddressFamily();  
9 [VB] Overrides Public ReadOnly Property AddressFamily As AddressFamily  
10 [JScript] public function get AddressFamily() : AddressFamily;

11  
12 *Description*  
13     Gets the Internet Protocol (IP) address family.  
14     Port  
15     ToString  
16

17 [C#] public int Port {get; set;}  
18 [C++] public: \_\_property int get\_Port();public: \_\_property void set\_Port(int);  
19 [VB] Public Property Port As Integer  
20 [JScript] public function get Port() : int;public function set Port(int);

21  
22 *Description*  
23     Gets or sets the TCP port number of the end point.  
24     Create  
25

1  
2 [C#] public override EndPoint Create(SocketAddress socketAddress);  
3 [C++] public: EndPoint\* Create(SocketAddress\* socketAddress);  
4 [VB] Overrides Public Function Create( ByVal socketAddress As SocketAddress)  
5 As EndPoint  
6 [JScript] public override function Create(socketAddress : SocketAddress) :  
7 EndPoint;

8  
9 *Description*

10 Creates an end point from a socket address.

11 *Return Value:* An **System.Net.EndPoint** instance using the specified socket. The  
12 network socket to use for the end point.

13 Equals

14  
15 [C#] public override bool Equals(object comparand);  
16 [C++] public: bool Equals(Object\* comparand);  
17 [VB] Overrides Public Function Equals( ByVal comparand As Object) As Boolean  
18 [JScript] public override function Equals(comparand : Object) : Boolean;

19 GetHashCode

20  
21 [C#] public override int GetHashCode();  
22 [C++] public: int GetHashCode();  
23 [VB] Overrides Public Function GetHashCode() As Integer  
24 [JScript] public override function GetHashCode() : int;

25 Serialize

[C#] public override SocketAddress Serialize();  
[C++] public: SocketAddress\* Serialize();  
[VB] Overrides Public Function Serialize() As SocketAddress  
[JScript] public override function Serialize() : SocketAddress;

### *Description*

Serializes end point information into a **System.Net.SocketAddress** instance.

*Return Value:* A **System.Net.SocketAddress** instance containing the socket address for the end point.

## ToString

[C#] public override string ToString();  
[C++] public: String\* ToString();  
[VB] Overrides Public Function ToString() As String  
[JScript] public override function ToString() : String;

### *Description*

Returns the IP address and port number for the specified end point.

*Return Value:* A string containing the IP address, in dotted-quad notation, and the port number for the specified end point (for example, 192.168.1.2:23).

## IPHostEntry class (System.Net)

ToString

1  
2  
3 *Description*  
4 Provides a container class for Internet host address information.  
5  
6 The **System.Net.IPHostEntry** class associates a Domain Name System  
(DNS) host name with an array of aliases and an array of matching IP addresses.

7 IPHostEntry

8 *Example Syntax:*

9 ToString

10  
11 [C#] public IPHostEntry();  
12 [C++] public: IPHostEntry();  
13 [VB] Public Sub New()  
14 [JScript] public function IPHostEntry();

15 AddressList

16 ToString

17  
18 [C#] public IPAddress[] AddressList {get; set;}  
19 [C++] public: \_\_property IPAddress\* get\_AddressList();public: \_\_property void  
20 set\_AddressList(IPAddress\*[]);  
21 [VB] Public Property AddressList As IPAddress ()  
22 [JScript] public function get AddressList() : IPAddress[];public function set  
23 AddressList(IPAddress[]);

24  
25 *Description*

1       Gets or sets a list of IP addresses associated with a host.

2       Aliases

3       ToString

4

5       [C#] public string[] Aliases {get; set;}

6       [C++] public: \_\_property String\* get\_Aliases();public: \_\_property void

7       set\_Aliases(String\* \_\_gc[]);

8       [VB] Public Property Aliases As String ()

9       [JScript] public function get Aliases() : String[];public function set

10      Aliases(String[]);

11

12      *Description*

13       Gets or sets a list of aliases associated with a host.

14       HostName

15       ToString

16

17       [C#] public string HostName {get; set;}

18       [C++] public: \_\_property String\* get\_HostName();public: \_\_property void

19       set\_HostName(String\*);

20       [VB] Public Property HostName As String

21       [JScript] public function get HostName() : String;public function set

22       HostName(String);

23

24      *Description*

25       Gets or sets the DNS name of the host.

1        The **System.Net.IPHostEntry.HostName** property contains the primary  
2 host name for a server. If the DNS entry for the server defines additional aliases,  
3 they will be available in the **System.Net.IPHostEntry.Aliases** property.

4        IWebProxy interface (System.Net)

5        ToString

6

7        *Description*

8

9        Provides the base interface for implementation of proxy access for the  
10      **System.Net.WebRequest** class.

11

12      The **System.Net.IWebProxy** interface provides the methods and properties  
13      required by the **System.Net.WebRequest** class to access proxy servers.

14        Credentials

15

16        ToString

17

18        [C#] ICredentials Credentials {get; set;}

19        [C++] ICredentials\* get\_Credentials();void set\_Credentials(ICredentials\*);

20        [VB] Property Credentials As ICredentials

21

22        [JScript] abstract function get Credentials() : ICredentials;public abstract function  
23        set Credentials(ICredentials);

24

25        *Description*

26        The credentials to submit to the proxy server for authentication.

1        The **System.Net.IWebProxy.Credentials** property is an  
2        **System.Net.ICredentials** instance containing the authorization credentials to send  
3        to the proxy server in response to an HTTP 407 (proxy authorization) status code.

4        **GetProxy**

5  
6        [C#] Uri GetProxy(Uri destination);  
7        [C++] Uri\* GetProxy(Uri\* destination);  
8        [VB] Function GetProxy(ByName destination As Uri) As Uri  
9        [JScript] function GetProxy(destination : Uri) : Uri;

10  
11      *Description*

12        Returns the URI of a proxy.

13      *Return Value:* A **System.Uri** containing the URI of the proxy used to contact  
14      *destination* .

15      The **System.Net.IWebProxy.GetProxy(System.Uri)** method returns the  
16      URI of the proxy server that handles requests to the Internet resource specified in  
17      the *destination* parameter. A **System.Uri** specifying the requested Internet  
18      resource.

19      **IsBypassed**

20  
21      [C#] bool IsBypassed(Uri host);  
22      [C++] bool IsBypassed(Uri\* host);  
23      [VB] Function IsBypassed(ByName host As Uri) As Boolean  
24      [JScript] function IsBypassed(host : Uri) : Boolean;

1  
2 *Description*

3       Indicates that the proxy should not be used for the specified host.

4       *Return Value*: **true** if the proxy server should not be used for *host* ; otherwise,

5       **false** .

6       The **System.Net.IWebProxy.IsBypassed(System.Uri)** method indicates  
7       whether to use the proxy server to access the host specified in the *host* parameter.  
8       If **System.Net.IWebProxy.IsBypassed(System.Uri)** is **true** , the proxy is not  
9       used to contact the host and the request is passed directly to the server. The  
10      **System.Uri** of the host to check for proxy use.

11      IWebRequestCreate interface (System.Net)

12      IsBypassed

13  
14  
15 *Description*

16       Provides the base interface for creating **System.Net.WebRequest**  
17       instances.

18       The **System.Net.IWebRequestCreate** interface defines the method that  
19       **System.Net.WebRequest** descendants must use to register with the  
20       **System.Net.WebRequest.Create(System.Uri,System.Boolean)** method.

21      Create

22  
23 [C#] WebRequest Create(Uri uri);

24 [C++] WebRequest\* Create(Uri\* uri);

25 [VB] Function Create(ByVal uri As Uri) As WebRequest

1 [JScript] function Create(uri : Uri) : WebRequest;

3 *Description*

4 Creates a **System.Net.WebRequest** instance.

5 *Return Value:* A **System.Net.WebRequest** instance.

6 The **System.Net.IWebRequestCreate.Create(System.Uri)** method must  
7 return an initialized instance of the **System.Net.WebRequest** descendant capable  
8 of performing a standard request/response transaction for the protocol without  
9 needing any protocol-specific fields modified. The uniform resource identifier  
10 (URI) of the Web resource.

11 NetworkAccess enumeration (System.Net)

12 Create

15 *Description*

16 Specifies network access permissions.

17 The **System.Net.NetworkAccess** enumeration is used with the  
18 **System.Net.WebPermission** and **System.Net.SocketPermission** classes.

19 Create

21 [C#] public const NetworkAccess Accept;

22 [C++] public: const NetworkAccess Accept;

23 [VB] Public Const Accept As NetworkAccess

24 [JScript] public var Accept : NetworkAccess;

1           *Description*

2           Indicates that the application is allowed to accept connections from the  
3           Internet on a local resource.

4           Create

5  
6  
7           [C#] public const NetworkAccess Connect;  
8           [C++] public: const NetworkAccess Connect;  
9           [VB] Public Const Connect As NetworkAccess  
10          [JS] public var Connect : NetworkAccess;

11  
12          *Description*

13          Indicates that the application is allowed to connect to specific Internet  
14          resources.

15          NetworkCredential class (System.Net)

16          ToString

17  
18  
19          *Description*

20          Provides credentials for password-based authentication schemes such as  
21          basic, digest, NTLM, and Kerberos authentication.

22          The **System.Net.NetworkCredential** class is a base class that supplies  
23          credentials in password-based authentication schemes such as basic, digest,  
24          NTLM, and Kerberos. Classes that implement the **System.Net.ICredentials**

1 interface, such as the **System.Net.CredentialCache** class, return  
2 **System.Net.NetworkCredential** instances.

3 NetworkCredential

4 *Example Syntax:*

5 ToString

6

7 [C#] public NetworkCredential();

8 [C++] public: NetworkCredential();

9 [VB] Public Sub New()

10 [JScript] public function NetworkCredential();

11 NetworkCredential

12 *Example Syntax:*

13 ToString

14

15 [C#] public NetworkCredential(string userName, string password);

16 [C++] public: NetworkCredential(String\* userName, String\* password);

17 [VB] Public Sub New(ByVal userName As String, ByVal password As String)

18 [JScript] public function NetworkCredential(userName : String, password :

19 String); Initializes a new instance of the **System.Net.NetworkCredential** class.

21 *Description*

22 Initializes a new instance of the **System.Net.NetworkCredential** class with  
23 the specified user name and password.

24 The constructor initializes a **System.Net.NetworkCredential** instance is  
25 initialized with the **System.Net.NetworkCredential.UserName** property set to

1    *userName* and the **System.Net.NetworkCredential.Password** property set to  
2    *password* . The user name associated with the credentials. The password for the  
3    user name associated with the credentials.

4            **NetworkCredential**

5            *Example Syntax:*

6            **ToString**

7  
8    [C#] public NetworkCredential(string *userName*, string *password*, string *domain*);

9    [C++] public: NetworkCredential(String\* *userName*, String\* *password*, String\*  
10    *domain*);

11    [VB] Public Sub New(ByVal *userName* As String, ByVal *password* As String,  
12        ByVal *domain* As String)

13    [JScript] public function NetworkCredential(*userName* : String, *password* : String,  
14        *domain* : String);

15  
16    *Description*

17        Initializes a new instance of the **System.Net.NetworkCredential** class with  
18        the specified user name, password, and domain.

19        The constructor initializes a **System.Net.NetworkCredential** instance with  
20        the **System.Net.NetworkCredential.UserName** property set to *userName* , the  
21        **System.Net.NetworkCredential.Password** property set to *password* , and the  
22        **System.Net.NetworkCredential.Domain** property set to *domain* . The user name  
23        associated with the credentials. The password for the user name associated with  
24        the credentials. The domain associated with these credentials.

25            **Domain**

1       ToString

2

3 [C#] public string Domain {get; set;}

4 [C++] public: \_\_property String\* get\_Domain();public: \_\_property void

5 set\_Domain(String\*);

6 [VB] Public Property Domain As String

7 [JScript] public function get Domain() : String;public function set Domain(String);

8

9 *Description*

10       Gets or sets the domain or machine name that verifies the credentials.

11       The **System.Net.NetworkCredential.Domain** property specifies the  
12 domain or realm to which the user name belongs. Typically, this is the host  
13 machine name where the application runs or user domain for the currently logged  
14 in user.

15       Password

16       ToString

17

18 [C#] public string Password {get; set;}

19 [C++] public: \_\_property String\* get\_Password();public: \_\_property void

20 set\_Password(String\*);

21 [VB] Public Property Password As String

22 [JScript] public function get Password() : String;public function set

23 Password(String);

24

25 *Description*

Gets or sets the password for the user name associated with the credentials.

UserName

ToString

[C#] public string UserName {get; set;}

[C++] public: \_\_property String\* get\_UserName();public: \_\_property void set\_UserName(String\*);

[VB] Public Property UserName As String

[JScript] public function get UserName() : String;public function set UserName(String);

### Description

Gets or sets the user name associated with the credentials.

## GetCredential

```
[C#] public NetworkCredential GetCredential(Uri uri, string authType);  
  
[C++] public: __sealed NetworkCredential* GetCredential(Uri* uri, String*  
authType);  
  
[VB] NotOverridable Public Function GetCredential(ByVal uri As Uri, ByVal  
authType As String) As NetworkCredential  
  
[JScript] public function GetCredential(uri : Uri, authType : String) :  
NetworkCredential;
```

### *Description*

1        Returns an instance of the **System.Net.NetworkCredential** class for the  
2        specified URI and authentication type.

3        *Return Value:* A **System.Net.NetworkCredential** instance. The URI that the  
4        client is providing authentication for. The type of authentication requested as  
5        defined in the **System.Net.IAuthenticationModule.AuthenticationType**  
6        property.

7        **ProtocolViolationException** class (System.Net)

8        **ToString**

9  
10  
11        *Description*

12        The exception that is thrown when an error is made while using a network  
13        protocol.

14        A **System.Net.ProtocolViolationException** is thrown by descendants of  
15        **System.Net.WebRequest** and **System.Net.WebResponse** to indicate an error  
16        using the underlying protocol. For example, the **System.Net.HttpWebRequest**  
17        and **System.Net.HttpWebResponse** classes throw a  
18        **System.Net.ProtocolViolationException** to indicate an error using HTTP.

19        **ProtocolViolationException**

20        *Example Syntax:*

21        **ToString**

22  
23        [C#] public ProtocolViolationException();

24        [C++] public: ProtocolViolationException();

25        [VB] Public Sub New()

1 [JScript] public function ProtocolViolationException(); Initializes a new instance  
2 of the **System.Net.ProtocolViolationException** class.

3

4 *Description*

5       Initializes a new instance of the **System.Net.ProtocolViolationException**  
6 class.

7       The default constructor initializes a new instance of the  
8 **System.Net.ProtocolViolationException** class with all properties set to **null**.

9       ProtocolViolationException

10      *Example Syntax:*

11      ToString

12

13 [C#] public ProtocolViolationException(string message);

14 [C++] public: ProtocolViolationException(String\* message);

15 [VB] Public Sub New(ByVal message As String)

16 [JScript] public function ProtocolViolationException(message : String);

17

18 *Description*

19       Initializes a new instance of the **System.Net.ProtocolViolationException**  
20 class with the specified message.

21       The **System.Net.ProtocolViolationException.#ctor** constructor initializes  
22 a new instance of the **System.Net.ProtocolViolationException** class with the  
23 **System.Exception.Message** property set to the value of the *message* parameter.

24       The error message string.

25       ProtocolViolationException

1       *Example Syntax:*

2        ToString

3

4       [C#] protected ProtocolViolationException(SerializationInfo serializationInfo,

5       StreamingContext streamingContext);

6       [C++] protected: ProtocolViolationException(SerializationInfo\* serializationInfo,

7       StreamingContext streamingContext);

8       [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal

9       streamingContext As StreamingContext)

10       [JScript] protected function ProtocolViolationException(serializationInfo :

11       SerializationInfo, streamingContext : StreamingContext);

12           HelpLink

13           HResult

14           InnerException

15           Message

16           Source

17           StackTrace

18           TargetSite

19           ISerializable.GetObjectData

20

21       [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,

22       StreamingContext streamingContext);

23       [C++] void ISerializable::GetObjectData(SerializationInfo\* serializationInfo,

24       StreamingContext streamingContext);

25       [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal

```
1 streamingContext As StreamingContext) Implements ISerializable.GetObjectData
2 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,
3 streamingContext : StreamingContext);
4
5 ServicePoint class (System.Net)
6
7 ToString
```

#### 8 *Description*

9 Provides connection management for HTTP connections.

10 The **System.Net.ServicePoint** class handles connections to an Internet  
11 resource based on the host information passed in the resource's URI. The initial  
12 connection to the resource determines the information the  
13 **System.Net.ServicePoint** maintains, which is then shared by all subsequent  
14 requests to that resource.

15 Address

16 ToString

```
17
18 [C#] public Uri Address {get;}
19 [C++] public: __property Uri* get_Address();
20 [VB] Public ReadOnly Property Address As Uri
21 [JScript] public function get Address() : Uri;
```

#### 23 *Description*

24 Gets the URI of the **System.Net.ServicePoint**.

25 Certificate

```
1    ToString  
2  
3 [C#] public X509Certificate Certificate {get;}  
4 [C++] public: __property X509Certificate* get_Certificate();  
5 [VB] Public ReadOnly Property Certificate As X509Certificate  
6 [JScript] public function get Certificate() : X509Certificate;  
7  
8 Description  
9     Gets the certificate received for this System.Net.ServicePoint .  
10    Although a System.Net.ServicePoint can make multiple connections to an  
11 Internet resource, it can maintain only one certificate.  
12    ClientCertificate  
13    ToString  
14  
15 [C#] public X509Certificate ClientCertificate {get;}  
16 [C++] public: __property X509Certificate* get_ClientCertificate();  
17 [VB] Public ReadOnly Property ClientCertificate As X509Certificate  
18 [JScript] public function get ClientCertificate() : X509Certificate; Gets the Client  
19 Certificate sent by us to the Server.  
20    ConnectionLimit  
21    ToString  
22  
23 [C#] public int ConnectionLimit {get; set;}  
24 [C++] public: __property int get_ConnectionLimit(); public: __property void  
25 set_ConnectionLimit(int);
```

```
1 [VB] Public Property ConnectionLimit As Integer  
2 [JScript] public function get ConnectionLimit() : int;public function set  
3 ConnectionLimit(int);  
4
```

```
5 Description
```

```
6 Gets or sets the maximum number of connections allowed on this  
7 System.Net.ServicePoint .
```

```
8 The System.Net.ServicePoint.ConnectionLimit property sets the  
9 maximum number of connections that the System.Net.ServicePoint can make to  
10 an Internet resource. The value of the System.Net.ServicePoint.ConnectionLimit  
11 property is set to the value of the  
12 System.Net.ServicePointManager.DefaultConnectionLimit property when the  
13 System.Net.ServicePoint is created; subsequent changes to  
14 System.Net.ServicePointManager.DefaultConnectionLimit have no effect on  
15 existing System.Net.ServicePoint instances.
```

```
16 ConnectionName
```

```
17 ToString
```

```
18  
19 [C#] public string ConnectionName {get;}  
20 [C++] public: __property String* get_ConnectionName();  
21 [VB] Public ReadOnly Property ConnectionName As String  
22 [JScript] public function get ConnectionName() : String;
```

```
23  
24 Description
```

1       Gets the connection group name established by the  
2 **System.Net.WebRequest** that created the connection.  
3       The **System.Net.ServicePoint.ConnectionName** property contains the  
4 connection group assigned to the  
5 **System.Net.WebRequest.ConnectionGroupName** property of the  
6 **System.Net.WebRequest** that initiated the connection provided by this  
7 **System.Net.ServicePoint**. If the **System.Net.ServicePoint.ConnectionName**  
8 property is set, only **System.Net.WebRequest** instances with the same  
9 **System.Net.WebRequest.ConnectionGroupName** can use this  
10 **System.Net.ServicePoint**.

11      CurrentConnections

12      ToString

13  
14 [C#] public int CurrentConnections {get;}  
15 [C++] public: \_\_property int get\_CurrentConnections();  
16 [VB] Public ReadOnly Property CurrentConnections As Integer  
17 [JScript] public function get CurrentConnections() : int;

18  
19 *Description*

20       Gets the number of connections associated with this  
21 **System.Net.ServicePoint**.

22       The **System.Net.ServicePoint.CurrentConnections** property contains the  
23 number of active Internet connections associated with this  
24 **System.Net.ServicePoint**. The value of

1   **System.Net.ServicePoint.CurrentConnections** cannot exceed that of  
2   **System.Net.ServicePoint.ConnectionLimit** .

3   IdleSince

4   ToString

6   [C#] public DateTime IdleSince {get;}

7   [C++] public: \_\_property DateTime get\_IdleSince();

8   [VB] Public ReadOnly Property IdleSince As DateTime

9   [JScript] public function get IdleSince() : DateTime;

11   *Description*

12       Gets the date and time that the **System.Net.ServicePoint** was last accessed.

13       The **System.Net.ServicePoint.IdleSince** property records the last date and  
14       time at which a service point was accessed. When the difference between the  
15       current time and **System.Net.ServicePoint.IdleSince** exceeds the value of  
16       **System.Net.ServicePoint.MaxIdleTime** , the **System.Net.ServicePoint** is  
17       available for recycling to another connection.

18   MaxIdleTime

19   ToString

21   [C#] public int MaxIdleTime {get; set;}

22   [C++] public: \_\_property int get\_MaxIdleTime();public: \_\_property void  
23   set\_MaxIdleTime(int);

24   [VB] Public Property MaxIdleTime As Integer

25   [JScript] public function get MaxIdleTime() : int;public function set

1 MaxIdleTime(int);

3 *Description*

4 Gets or sets the maximum idle time for the **System.Net.ServicePoint** .

5 The **System.Net.ServicePoint.MaxIdleTime** property contains the length  
6 of time, in milliseconds, that the **System.Net.ServicePoint** is allowed to maintain  
7 an idle connection to an Internet resource before it is recycled for use in another  
8 connection.

9 ProtocolVersion

10 ToString

12 [C#] public virtual Version ProtocolVersion {get;}

13 [C++] public: \_\_property virtual Version\* get\_ProtocolVersion();

14 [VB] Overridable Public ReadOnly Property ProtocolVersion As Version

15 [JScript] public function get ProtocolVersion() : Version;

17 *Description*

18 Gets the version of the HTTP protocol that the **System.Net.ServicePoint**  
19 uses.

20 SupportsPipelining

21 ToString

23 [C#] public bool SupportsPipelining {get;}

24 [C++] public: \_\_property bool get\_SupportsPipelining();

25 [VB] Public ReadOnly Property SupportsPipelining As Boolean

1 [JScript] public function get SupportsPipelining() : Boolean;

3 *Description*

4     Indicates whether the **System.Net.ServicePoint** supports pipelined  
5     connections.

6     GetHashCode

8 [C#] public override int GetHashCode();

9 [C++] public: int GetHashCode();

10 [VB] Overrides Public Function GetHashCode() As Integer

11 [JScript] public override function GetHashCode() : int;

13 *Description*

14     Gets the hash code for the **System.Net.ServicePoint** .

15 *Return Value:* The hash code for the **System.Net.ServicePoint** .

16     The hash code for **System.Net.ServicePoint** A and B is the same if  
17     A.Equals(B) is **true** .

18     ServicePointManager class (System.Net)

19     ToString

22 *Description*

23     Manages the collection of **System.Net.ServicePoint** instances.

24     **System.Net.ServicePointManager** is a static class used to create,  
25     maintain, and delete instances of the **System.Net.ServicePoint** class.

```
1    ToString  
2  
3 [C#] public const int DefaultNonPersistentConnectionLimit;  
4 [C++] public: const int DefaultNonPersistentConnectionLimit;  
5 [VB] Public Const DefaultNonPersistentConnectionLimit As Integer  
6 [JScript] public var DefaultNonPersistentConnectionLimit : int;  
7  
8 Description  
9 The default number of nonpersistent connections allowed on a  
10 System.Net.ServicePoint connected to an HTTP/1.1 server.
```

```
11   ToString  
12  
13 [C#] public const int DefaultPersistentConnectionLimit;  
14 [C++] public: const int DefaultPersistentConnectionLimit;  
15 [VB] Public Const DefaultPersistentConnectionLimit As Integer  
16 [JScript] public var DefaultPersistentConnectionLimit : int;  
17  
18 Description  
19 The default number of persistent connections allowed on a  
20 System.Net.ServicePoint connected to an HTTP/1.0 server.
```

```
21   CertificatePolicy
```

```
22   ToString  
23  
24 [C#] public static ICertificatePolicy CertificatePolicy {get; set;}  
25 [C++] public: __property static ICertificatePolicy* get_CertificatePolicy(); public:
```

```
1  __property static void set_CertificatePolicy(ICertificatePolicy*);  
2  [VB] Public Shared Property CertificatePolicy As ICertificatePolicy  
3  [JScript] public static function get CertificatePolicy() : ICertificatePolicy;public  
4  static function set CertificatePolicy(ICertificatePolicy);  
5  
6  Description
```

7       Gets or sets policy for server certificates.

8       When the **System.Net.ServicePointManager.CertificatePolicy** property is  
9       set to an **System.Net.ICertificatePolicy** interface instance, the  
10      **System.Net.ServicePointManager** uses the certificate policy defined in that  
11      instance instead of the default certificate policy.

12     DefaultConnectionLimit

13     ToString

14  
15 [C#] public static int DefaultConnectionLimit {get; set;}

16 [C++] public: \_\_property static int get\_DefaultConnectionLimit();public:  
17 \_\_property static void set\_DefaultConnectionLimit(int);

18 [VB] Public Shared Property DefaultConnectionLimit As Integer

19 [JScript] public static function get DefaultConnectionLimit() : int;public static  
20 function set DefaultConnectionLimit(int);

21  
22 *Description*

23       The maximum number of concurrent connections allowed by a

24      **System.Net.ServicePoint** instance.

1        The **System.Net.ServicePointManager.DefaultConnectionLimit**  
2 property sets the default maximum number of concurrent connections that the  
3 **System.Net.ServicePointManager** assigns to the  
4 **System.Net.ServicePoint.ConnectionLimit** property when creating  
5 **System.Net.ServicePoint** instances.

6        **MaxServicePointIdleTime**

7        **ToString**

8  
9 [C#] public static int MaxServicePointIdleTime {get; set;}  
10 [C++] public: \_\_property static int get\_MaxServicePointIdleTime();public:  
11        \_\_property void set\_MaxServicePointIdleTime(int);  
12 [VB] Public Shared Property MaxServicePointIdleTime As Integer  
13 [JScript] public static function get MaxServicePointIdleTime() : int;public static  
14 function set MaxServicePointIdleTime(int);  
15

16 *Description*

17        Gets or sets the maximum idle time of a **System.Net.ServicePoint**  
18 instance.

19        The **System.Net.ServicePointManager.MaxServicePointIdleTime**  
20 property sets the maximum idle time that the **System.Net.ServicePointManager**  
21 assigns to the **System.Net.ServicePoint.MaxIdleTime** property when creating  
22 **System.Net.ServicePoint** instances. Changes to this value will affect only  
23 **System.Net.ServicePoint** instances that are initialized after the value is changed.

24        **MaxServicePoints**

25        **ToString**

```
1  
2 [C#] public static int MaxServicePoints {get; set;}  
3 [C++] public: __property static int get_MaxServicePoints();public: __property  
4 static void set_MaxServicePoints(int);  
5 [VB] Public Shared Property MaxServicePoints As Integer  
6 [JScript] public static function get MaxServicePoints() : int;public static function  
7 set MaxServicePoints(int);  
8
```

#### 9 *Description*

10 Gets or sets the maximum number of **System.Net.ServicePoint** instances  
11 to maintain at any time.

12 When you reduce the  
13 **System.Net.ServicePointManager.MaxServicePoints** property below the  
14 number of **System.Net.ServicePoint** instances currently in existence, the  
15 **System.Net.ServicePointManager** deletes the **System.Net.ServicePoint**  
16 instances with the longest idle times. If the number of **System.Net.ServicePoint**  
17 instances with active connections is greater than the value of  
18 **System.Net.ServicePointManager.MaxServicePoints** , the  
19 **System.Net.ServicePointManager** will delete the **System.Net.ServicePoint**  
20 instances as they become idle.

#### 21 *FindServicePoint*

```
22  
23 [C#] public static ServicePoint FindServicePoint(Uri address);  
24 [C++] public: static ServicePoint* FindServicePoint(Uri* address);  
25 [VB] Public Shared Function FindServicePoint(ByVal address As Uri) As
```

1 ServicePoint  
2 [JScript] public static function FindServicePoint(address : Uri) : ServicePoint;  
3 Finds an existing **System.Net.ServicePoint** or creates a new  
4 **System.Net.ServicePoint** to manage communication for this request.

5  
6 *Description*

7 Finds an existing **System.Net.ServicePoint** or creates a new  
8 **System.Net.ServicePoint** to manage communications with the specified  
9 **System.Uri** .

10 *Return Value*: The **System.Net.ServicePoint** that manages communications for  
11 the request.

12 The **System.Net.ServicePointManager.FindServicePoint(System.Uri)**  
13 method returns the **System.Net.ServicePoint** instance associated with the  
14 specified Internet host name. If no **System.Net.ServicePoint** exists for that host,  
15 the **System.Net.ServicePointManager** creates one. The **System.Uri** of the  
16 Internet resource to contact.

17 FindServicePoint

18  
19 [C#] public static ServicePoint FindServicePoint(string uriString, IWebProxy  
20 proxy);

21 [C++] public: static ServicePoint\* FindServicePoint(String\* uriString,  
22 IWebProxy\* proxy);

23 [VB] Public Shared Function FindServicePoint(ByVal uriString As String, ByVal  
24 proxy As IWebProxy) As ServicePoint

25 [JScript] public static function FindServicePoint(uriString : String, proxy :

1 IWebProxy) : ServicePoint; Finds an existing **System.Net.ServicePoint** or creates  
2 a new **System.Net.ServicePoint** to manage communication for this request.

3

4 *Description*

5 Finds an existing **System.Net.ServicePoint** or creates a new  
6 **System.Net.ServicePoint** to manage communications with the specified URI.

7 *Return Value*: The **System.Net.ServicePoint** that manages communications for  
8 the request.

9 The **System.Net.ServicePointManager.FindServicePoint(System.Uri)**  
10 method returns the **System.Net.ServicePoint** instance associated with the  
11 specified Internet host name. If no **System.Net.ServicePoint** exists for that host,  
12 the **System.Net.ServicePointManager** creates one. The URI of the Internet  
13 resource to be contacted. Proxy data for this request.

14 **FindServicePoint**

15

16 [C#] public static ServicePoint FindServicePoint(Uri address, IWebProxy proxy);  
17 [C++] public: static ServicePoint\* FindServicePoint(Uri\* address, IWebProxy\*  
18 proxy);

19 [VB] Public Shared Function FindServicePoint(ByVal address As Uri, ByVal  
20 proxy As IWebProxy) As ServicePoint

21 [JScript] public static function FindServicePoint(address : Uri, proxy :  
22 IWebProxy) : ServicePoint;

23

24 *Description*

1       Finds an existing **System.Net.ServicePoint** or creates a new  
2 **System.Net.ServicePoint** to manage communications with the specified  
3 **System.Uri** instance.

4 *Return Value:* The **System.Net.ServicePoint** that manages communications for  
5 the request.

6       The **System.Net.ServicePointManager.FindServicePoint(System.Uri)**  
7 method returns the **System.Net.ServicePoint** instance associated with the  
8 specified Internet host name. If no **System.Net.ServicePoint** exists for that host,  
9 the **System.Net.ServicePointManager** creates one. A **System.Uri** instance  
10 containing the address of the Internet resource to contact. Proxy data for this  
11 request.

12       **SocketAddress** class (System.Net)

13       **ToString**

16 *Description*

17       Identifies a socket address.

18       **SocketAddress**

19       *Example Syntax:*

20       **ToString**

22 [C#] public **SocketAddress**(**AddressFamily** family);

23 [C++] public: **SocketAddress**(**AddressFamily** family);

24 [VB] **Public Sub** New(**ByVal** family **As** **AddressFamily**)

25 [JScript] **public function** **SocketAddress**(family : **AddressFamily**); **Initializes a**

1 new instance of the **System.Net.SocketAddress** class.

3 *Description*

4       Initializes a new instance of the **System.Net.SocketAddress** class for the  
5       given address family.

6       SocketAddress

7       *Example Syntax:*

8       ToString

10      [C#] public SocketAddress(AddressFamily family, int size);

11      [C++] public: SocketAddress(AddressFamily family, int size);

12      [VB] Public Sub New(ByVal family As AddressFamily, ByVal size As Integer)

13      [JScript] public function SocketAddress(family : AddressFamily, size : int);

15 *Description*

17       Family

18       ToString

20      [C#] public AddressFamily Family {get;}

21      [C++] public: \_\_property AddressFamily get\_Family();

22      [VB] Public ReadOnly Property Family As AddressFamily

23      [JScript] public function get Family() : AddressFamily;

25 *Description*

1  
2       Item  
3       ToString  
4  
5 [C#] public byte this[int offset] {get; set;}  
6 [C++] public: \_\_property unsigned char get\_Item(int offset);public: \_\_property  
7 void set\_Item(int offset, unsigned char);  
8 [VB] Public Default Property Item(ByVal offset As Integer) As Byte  
9 [JScript] returnValue =  
10      SocketAddressObject.Item(offset);SocketAddressObject.Item(offset) =  
11      returnValue;

12  
13 *Description*

14  
15       Size  
16       ToString  
17  
18 [C#] public int Size {get;}  
19 [C++] public: \_\_property int get\_Size();  
20 [VB] Public ReadOnly Property Size As Integer  
21 [JScript] public function get Size() : int;

22  
23 *Description*

24  
25       Equals

1  
2 [C#] public override bool Equals(object comparand);  
3 [C++] public: bool Equals(Object\* comparand);  
4 [VB] Overrides Public Function Equals(ByVal comparand As Object) As Boolean  
5 [JScript] public override function Equals(comparand : Object) : Boolean;  
6       GetHashCode  
7  
8 [C#] public override int GetHashCode();  
9 [C++] public: int GetHashCode();  
10 [VB] Overrides Public Function GetHashCode() As Integer  
11 [JScript] public override function GetHashCode() : int;  
12       ToString  
13  
14 [C#] public override string ToString();  
15 [C++] public: String\* ToString();  
16 [VB] Overrides Public Function ToString() As String  
17 [JScript] public override function ToString() : String;  
18       SocketPermission class (System.Net)  
19       ToString  
20  
21  
22 *Description*  
23       Controls rights to make or accept connections on a transport address.  
24       ToString  
25

```
1  
2 [C#] public const int AllPorts;  
3 [C++] public: const int AllPorts;  
4 [VB] Public Const AllPorts As Integer  
5 [JScript] public var AllPorts : int;
```

7 *Description*

8     Defines a constant representing all ports.

9     **SocketPermission**

10    *Example Syntax:*

11    **ToString**

```
12  
13 [C#] public SocketPermission(PermissionState state);  
14 [C++] public: SocketPermission(PermissionState state);  
15 [VB] Public Sub New(ByVal state As PermissionState)  
16 [JScript] public function SocketPermission(state : PermissionState); Initializes a  
17 new instance of the System.Net.SocketPermission class.
```

19 *Description*

20    Initializes a new instance of the **System.Net.SocketPermission** class that  
21 passes all demands or fails all demands.

22    If the **System.Net.SocketPermission** instance was created with the  
23 **Unrestricted** value from **System.Security.Permissions.PermissionState** then the  
24 **System.Net.SocketPermission** instance will pass all demands. Any other value

1 for *state* will result in a **System.Net.SocketPermission** instance that will fail all  
2 demands. One of the **System.Security.Permissions.PermissionState** values.

3 **SocketPermission**

4 *Example Syntax:*

5 **ToString**

6  
7 [C#] public SocketPermission(NetworkAccess access, TransportType transport,  
8 string hostName, int portNumber);

9 [C++] public: SocketPermission(NetworkAccess access, TransportType transport,  
10 String\* hostName, int portNumber);

11 [VB] Public Sub New(ByVal access As NetworkAccess, ByVal transport As  
12 TransportType, ByVal hostName As String, ByVal portNumber As Integer)

13 [JScript] public function SocketPermission(access : NetworkAccess, transport :  
14 TransportType, hostName : String, portNumber : int);

15  
16 *Description*

17 Initializes a new instance of the **System.Net.SocketPermission** class for  
18 the given transport address with the specified permission. One of the  
19 **System.Net.NetworkAccess** values. One of the **System.Net.TransportType**  
20 values. The host name for the transport address. The port number for the transport  
21 address.

22 **AcceptList**

23 **ToString**

24  
25 [C#] public IEnumarator AcceptList {get;}

```
1 [C++] public: __property IEnumarator* get_AcceptList();
2 [VB] Public ReadOnly Property AcceptList As IEnumarator
3 [JScript] public function get AcceptList() : IEnumarator;
```

```
4
```

5 *Description*

6 Gets a list of **System.Net.EndpointPermission** instances identifying the  
7 endpoints that can be accepted under this permission instance.

```
8 ConnectList
```

```
9 ToString
```

```
10
```

```
11 [C#] public IEnumarator ConnectList {get;}
```

```
12 [C++] public: __property IEnumarator* get_ConnectList();
```

```
13 [VB] Public ReadOnly Property ConnectList As IEnumarator
```

```
14 [JScript] public function get ConnectList() : IEnumarator;
```

```
15
```

16 *Description*

17 Gets a list of **System.Net.EndpointPermission** instances identifying the  
18 endpoints that can be connected to under this permission instance.

```
19 AddPermission
```

```
20
```

```
21 [C#] public void AddPermission(NetworkAccess access, TransportType transport,
22 string hostName, int portNumber);
```

```
23 [C++] public: void AddPermission(NetworkAccess access, TransportType
24 transport, String* hostName, int portNumber);
```

```
25 [VB] Public Sub AddPermission(ByVal access As NetworkAccess, ByVal
```

1    transport As TransportType, ByVal hostName As String, ByVal portNumber As  
2    Integer)  
3    [JScript] public function AddPermission(access : NetworkAccess, transport :  
4    TransportType, hostName : String, portNumber : int);  
5

6    *Description*

7       Adds a permission to the set of permissions for a transport address.

8       Permissions are checked with a logical OR operation. One of the

9       **System.Net.NetworkAccess** values. One of the **System.Net.TransportType**  
10      values. The host name for the transport address. The port number for the transport  
11      address.

12      Copy

13  
14      [C#] public override IPermission Copy();

15      [C++] public: IPermission\* Copy();

16      [VB] Overrides Public Function Copy() As IPermission

17      [JScript] public override function Copy() : IPermission;

18  
19      *Description*

20       Creates a copy of a **System.Net.SocketPermission** instance.

21      *Return Value:* A new instance of the **System.Net.SocketPermission** class that is a  
22      copy of the current instance.

23      FromXml

24  
25      [C#] public override void FromXml(SecurityElement securityElement);

```
1 [C++] public: void FromXml(SecurityElement* securityElement);  
2 [VB] Overrides Public Sub FromXml(ByVal securityElement As  
3 SecurityElement)  
4 [JScript] public override function FromXml(securityElement : SecurityElement);  
5
```

#### 6 *Description*

7       Reconstructs a **System.Net.SocketPermission** instance for an XML  
8 encoding.

9       The  
10      **System.Net.SocketPermission.FromXml(System.Security.SecurityElement)**  
11     method reconstructs a **System.Net.SocketPermission** instance from an XML  
12     encoding defined by the **System.Security.SecurityElement** class. The XML  
13     encoding used to reconstruct the **System.Net.SocketPermission** instance.

#### 14      Intersect

```
15  
16 [C#] public override IPermission Intersect(IPermission target);  
17 [C++] public: IPermission* Intersect(IPPermission* target);  
18 [VB] Overrides Public Function Intersect(ByVal target As IPermission) As  
19 IPermission  
20 [JScript] public override function Intersect(target : IPermission) : IPermission;  
21
```

#### 22 *Description*

23       Returns the logical intersection between two  
24      **System.Net.SocketPermission** instances.

25      *Return Value:* The **System.Net.SocketPermission** instance that represents the

1 intersection of two **System.Net.SocketPermission** instances. The  
2 **System.Net.SocketPermission** instance to combine with the current instance.

3 **IsSubsetOf**

4

5 [C#] public override bool IsSubsetOf(IPermission target);

6 [C++] public: bool IsSubsetOf(IPermission\* target);

7 [VB] Overrides Public Function IsSubsetOf( ByVal target As IPermission) As  
8 Boolean

9 [JScript] public override function IsSubsetOf(target : IPermission) : Boolean;

10

11 *Description*

12       Compares two **System.Net.SocketPermission** instances.

13 *Return Value:* **false** if *target* is **null** ; otherwise, an exception is thrown.

14       The **System.Net.SocketPermission** class does not support the  
15 **System.Net.SocketPermission.IsSubsetOf(System.Security.IPermission)**  
16 method. The second **System.Net.SocketPermission** instance to compare.

17       **IsUnrestricted**

18

19 [C#] public bool IsUnrestricted();

20 [C++] public: \_\_sealed bool IsUnrestricted();

21 [VB] NotOverridable Public Function IsUnrestricted() As Boolean

22 [JScript] public function IsUnrestricted() : Boolean;

23

24 *Description*

1        Checks the overall permission state of the object.  
2 *Return Value:* **true** if the **System.Net.SocketPermission** instance was created  
3 with the **Unrestricted** value from **System.Security.Permissions.PermissionState**  
4 ; otherwise, **false** .

5        **ToXml**

6  
7 [C#] public override SecurityElement ToXml();  
8 [C++] public: SecurityElement\* ToXml();  
9 [VB] Overrides Public Function ToXml() As SecurityElement  
10 [JScript] public override function ToXml() : SecurityElement;

11  
12 *Description*

13        Creates an XML encoding of a **System.Net.SocketPermission** instance and  
14 its current state.  
15 *Return Value:* A **System.Security.SecurityElement** instance containing an XML-  
16 encoded representation of the **System.Net.SocketPermission** instance, including  
17 state information.

18        The **System.Net.SocketPermission.ToXml** method creates a  
19 **System.Security.SecurityElement** instance to encode a representation of the  
20 **System.Net.SocketPermission** instance, including state information, in XML.

21        **Union**

22  
23 [C#] public override IPermission Union(IPermission target);  
24 [C++] public: IPermission\* Union(IPPermission\* target);  
25 [VB] Overrides Public Function Union(ByVal target As IPermission) As

1      IPermission  
2      [JScript] public override function Union(target : IPermission) : IPermission;

3  
4      *Description*

5              Returns the logical union between two **System.Net.SocketPermission**  
6 instances.

7      *Return Value:* The **System.Net.SocketPermission** instance that represents the  
8 union of two **System.Net.SocketPermission** instances. The  
9 **System.Net.SocketPermission** instance to combine with the current instance.

10              SocketPermissionAttribute class (System.Net)  
11              Union

12  
13  
14      *Description*

15              Enables security actions for **System.Net.SocketPermission** to be applied to  
16 code using declarative security. This class cannot be inherited.

17              SocketPermissionAttribute

18      *Example Syntax:*

19              Union

21      [C#] public SocketPermissionAttribute(SecurityAction action);

22      [C++] public: SocketPermissionAttribute(SecurityAction action);

23      [VB] Public Sub New(ByVal action As SecurityAction)

24      [JScript] public function SocketPermissionAttribute(action : SecurityAction);

1           *Description*

2           Initializes a new instance of the **System.Net.SocketPermissionAttribute**  
3           class with the specified **System.Security.Permissions.SecurityAction** value. One  
4           of the **System.Security.Permissions.SecurityAction** values.

5           Access

6           Union

7

8           [C#] public string Access {get; set;}

9

10          [C++] public: \_\_property String\* get\_Access(); public: \_\_property void  
11           set\_Access(String\*);

12          [VB] Public Property Access As String

13          [JScript] public function get Access() : String; public function set Access(String);

14

15           *Description*

16           Gets or sets the network access method allowed by this permission  
17           instance.

18           Action

19           Host

20           Union

21

22

23           *Description*

24           The DNS host name or IP address associated with this permission instance.

25           Port

1           Union  
2  
3 [C#] public string Port {get; set;}  
4 [C++] public: \_\_property String\* get\_Port();public: \_\_property void  
5 set\_Port(String\*);  
6 [VB] Public Property Port As String  
7 [JScript] public function get Port() : String;public function set Port(String);  
8

9           *Description*

10           Gets or sets the transport type associated with this permission instance.

11           Transport

12           Union

13  
14 [C#] public string Transport {get; set;}  
15 [C++] public: \_\_property String\* get\_Transport();public: \_\_property void  
16 set\_Transport(String\*);  
17 [VB] Public Property Transport As String  
18 [JScript] public function get Transport() : String;public function set  
19 Transport(String);  
20

21           *Description*

22           Gets or sets the transport type associated with this permission instance.

23           TypeId

24           Unrestricted

25           CreatePermission

1  
2 [C#] public override IPermission CreatePermission();  
3 [C++] public: IPermission\* CreatePermission();  
4 [VB] Overrides Public Function CreatePermission() As IPermission  
5 [JScript] public override function CreatePermission() : IPermission;  
6

7 *Description*

8       Creates and returns a new instance of the **System.Net.SocketPermission**  
9       class.

10      *Return Value:* An instance of the **System.Net.SocketPermission** class  
11     corresponding to the security declaration.

12      The **System.Net.SocketPermissionAttribute.CreatePermission** method is  
13     called by the security system, not by application code.

14      TransportType enumeration (System.Net)

15      ToString

16  
17 *Description*

18      Defines transport types for the **System.Net.SocketPermission** and  
19     **System.Net.Sockets.Socket** classes.

20      The **System.Net.TransportType** enumeration defines transport types for  
21     the **System.Net.SocketPermission** and **System.Net.Sockets.Socket** classes.

22  
23      ToString

24  
25 [C#] public const TransportType All;

1 [C++] public: const TransportType All;  
2 [VB] Public Const All As TransportType  
3 [JScript] public var All : TransportType;

5 *Description*

6 All transport types.

7 ToString

8  
9 [C#] public const TransportType Connectionless;  
10 [C++] public: const TransportType Connectionless;  
11 [VB] Public Const Connectionless As TransportType  
12 [JScript] public var Connectionless : TransportType;

14 *Description*

15 The transport type is connectionless, such as UDP.

16 ToString

17  
18 [C#] public const TransportType ConnectionOriented;  
19 [C++] public: const TransportType ConnectionOriented;  
20 [VB] Public Const ConnectionOriented As TransportType  
21 [JScript] public var ConnectionOriented : TransportType;

23 *Description*

24 The transport is connection oriented, such as TCP.

25 ToString

```
1  
2 [C#] public const TransportType Tcp;  
3 [C++] public: const TransportType Tcp;  
4 [VB] Public Const Tcp As TransportType  
5 [JScript] public var Tcp : TransportType;
```

7 *Description*

8     TCP transport.

9     ToString

```
10  
11 [C#] public const TransportType Udp;  
12 [C++] public: const TransportType Udp;  
13 [VB] Public Const Udp As TransportType  
14 [JScript] public var Udp : TransportType;
```

16 *Description*

17     UDP transport.

18     WebClient class (System.Net)

19     ToString

22 *Description*

23     Provides common methods for sending data to and receiving data from a  
24     resource identified by a URI. This class cannot be inherited.

1        The **System.Net.WebClient** class provides common methods for sending  
2 data to or receiving data from any local, Intranet, or Internet resource identified by  
3 a URI.

4        **WebClient**

5        *Example Syntax:*

6        **ToString**

7  
8        [C#] public WebClient();  
9        [C++] public: WebClient();  
10       [VB] Public Sub New()  
11       [JScript] public function WebClient();

12  
13       *Description*

14       Initializes a new instance of the **System.Net.WebClient** class.

15       The default constructor creates a new instance of the  
16       **System.Net.WebClient** class with all fields set to **null** .

17       **BaseAddress**

18       **ToString**

19  
20       [C#] public string BaseAddress {get; set;}  
21       [C++] public: \_\_property String\* get\_BaseAddress();public: \_\_property void  
22       set\_BaseAddress(String\*);  
23       [VB] Public Property BaseAddress As String  
24       [JScript] public function get BaseAddress() : String;public function set  
25       BaseAddress(String);

1  
2 *Description*

3 Gets or sets the base URI for requests made by a **System.Net.WebClient**.

4 The **System.Net.WebClient.BaseAddress** property contains a base URI  
5 that is combined with the relative address specified when calling an upload or  
6 download method.

7 Container

8 Credentials

9 ToString

10  
11  
12 *Description*

13 Gets or sets the network credentials used for authenticating the request with  
14 the Internet resource.

15 The **System.Net.WebClient.Credentials** property contains the  
16 authentication credentials required to access the Internet resource.

17 DesignMode

18 Events

19 Headers

20 ToString

21  
22  
23 *Description*

24 Gets or sets a collection of header name/value pairs associated with the  
25 request.

1        The **System.Net.WebClient.Headers** property contains a  
2        **System.Net.WebHeaderCollection** instance containing header information the  
3        **System.Net.WebClient** sends to the Internet resource. This is an unrestricted  
4        collection of headers, so setting headers that are restricted by  
5        **System.Net.WebRequest** descendants like **System.Net.HttpWebRequest** is  
6        allowed.

7        **QueryString**

8        **ToString**

9  
10      [C#] public NameValueCollection QueryString {get; set;}  
11      [C++] public: \_\_property NameValueCollection\* get\_QueryString();public:  
12      \_\_property void set\_QueryString(NameValueCollection\*);  
13      [VB] Public Property QueryString As NameValueCollection  
14      [JScript] public function get QueryString() : NameValueCollection;public function  
15      set QueryString(NameValueCollection);

16  
17      *Description*

18        Gets or sets a collection of query name/value pairs associated with the  
19        request.

20        The **System.Net.WebClient.QueryString** property contains a  
21        **System.Collections.Specialized.NameValueCollection** instance containing  
22        name/value pairs that are appended to the URI as a query string. The contents of  
23        the **System.Net.WebClient.QueryString** property are preceded by a question  
24        mark (?), and each name/value pair is separated by an ampersand (&).

25        **ResponseHeaders**

```
1    ToString  
2  
3    [C#] public WebHeaderCollection ResponseHeaders {get;}  
4    [C++] public: __property WebHeaderCollection* get_ResponseHeaders();  
5    [VB] Public ReadOnly Property ResponseHeaders As WebHeaderCollection  
6    [JScript] public function get ResponseHeaders() : WebHeaderCollection;  
7  
8 Description
```

Gets a collection of header name/value pairs associated with the response.

The **System.Net.WebClient.ResponseHeaders** property contains a **System.Net.WebHeaderCollection** instance containing header information the **System.Net.WebClient** receives from the Internet resource.

Site

DownloadData

```
15  
16    [C#] public byte[] DownloadData(string address);  
17    [C++] public: unsigned char DownloadData(String* address) __gc[];  
18    [VB] Public Function DownloadData(ByVal address As String) As Byte()  
19    [JScript] public function DownloadData(address : String) : Byte[];
```

```
20  
21 Description
```

Downloads data from a resource identified by a URI.

*Return Value:* A byte array containing the data downloaded from the resource specified by *address* .

1        The **System.Net.WebClient.DownloadData(System.String)** method  
2    downloads the data from the URI specified by *address* to a local byte array. The  
3    URI from which the data will be downloaded.

4        **DownloadFile**

5  
6        [C#] public void DownloadFile(string address, string fileName);  
7        [C++] public: void DownloadFile(String\* address, String\* fileName);  
8        [VB] Public Sub DownloadFile(ByVal address As String, ByVal fileName As  
9           String)  
10        [JScript] public function DownloadFile(address : String, fileName : String);

11  
12        *Description*

13        Downloads data from a resource identified by a URI to a local file.

14        The

15        **System.Net.WebClient.DownloadFile(System.String, System.String)** method  
16    downloads the data from the URI specified by *address* to a local file. The URI  
17    from which the data will be downloaded. The name of the local file to receive the  
18    data.

19        **OpenRead**

20  
21        [C#] public Stream OpenRead(string address);  
22        [C++] public: Stream\* OpenRead(String\* address);  
23        [VB] Public Function OpenRead(ByVal address As String) As Stream  
24        [JScript] public function OpenRead(address : String) : Stream;

1  
2 *Description*

3        Opens a readable stream for the data downloaded from a resource identified  
4 by a URI.

5 *Return Value:* A **System.IO.Stream** used to read data from a resource.

6        The **System.Net.WebClient.OpenRead(System.String)** method creates a  
7 **System.IO.Stream** instance used to access the data specified by *address* . The  
8 URI from which the data will be downloaded.

9        *OpenWrite*

10  
11 [C#] public Stream OpenWrite(string address);

12 [C++] public: Stream\* OpenWrite(String\* address);

13 [VB] Public Function OpenWrite(ByVal address As String) As Stream

14 [JScript] public function OpenWrite(address : String) : Stream; Opens a stream for  
15 writing data to a resource identified by a URI.

16  
17 *Description*

18        Opens a stream for writing data to the specified resource.

19 *Return Value:* A **System.IO.Stream** used to write data to the resource.

20        The **System.Net.WebClient.OpenWrite(System.String)** method returns a  
21 writable stream that is used to send data to a resource. The underlying request is  
22 made with the POST method. The URI of the resource to receive the data.

23        *OpenWrite*

24  
25 [C#] public Stream OpenWrite(string address, string method);

```
1 [C++] public: Stream* OpenWrite(String* address, String* method);  
2 [VB] Public Function OpenWrite(ByVal address As String, ByVal method As  
3 String) As Stream  
4 [JScript] public function OpenWrite(address : String, method : String) : Stream;
```

5  
6 *Description*

7       Opens a stream for writing data to the specified resource identified by a  
8       URI using the specified method.

9       *Return Value:* A **System.IO.Stream** used to write data to the resource.

10      The **System.Net.WebClient.OpenWrite(System.String)** method returns a  
11      writable stream that is used to send data to a resource. The underlying request is  
12      made with the method specified in the *method* parameter. The URI of the resource  
13      to receive the data. The method used to send the data to the resource.

14      UploadData

```
15  
16 [C#] public byte[] UploadData(string address, byte[] data);  
17 [C++] public: unsigned char UploadData(String* address, unsigned char data  
18        __gc[]) __gc[];  
19 [VB] Public Function UploadData(ByVal address As String, ByVal data() As  
20        Byte) As Byte()  
21 [JScript] public function UploadData(address : String, data : Byte[]) : Byte[];  
22      Uploads a data buffer to a resource identified by a URI.
```

23  
24 *Description*

1       Uploads a data buffer to the resource identified by a URI.

2       *Return Value:* An array of bytes containing the body of any response from the  
3       resource.

4       **The System.Net.WebClient.UploadData(System.String, System.Byte[])**

5       method sends a data buffer to a resource. The underlying request is made using the  
6       POST method. The URI of the resource to receive the data. The data buffer to  
7       send to the resource.

8       UploadData

9  
10      [C#] public byte[] UploadData(string address, string method, byte[] data);  
11      [C++] public: unsigned char UploadData(String\* address, String\* method,  
12            unsigned char data \_\_gc[]) \_\_gc[];  
13      [VB] Public Function UploadData(ByVal address As String, ByVal method As  
14            String, ByVal data() As Byte) As Byte()  
15      [JScript] public function UploadData(address : String, method : String, data :  
16            Byte[]) : Byte[];

17  
18      *Description*

19       Uploads a data buffer to the specified resource identified by a URI using  
20       the specified method.

21       *Return Value:* An array of bytes containing the body of any response from the  
22       resource.

23       **The System.Net.WebClient.UploadData(System.String, System.Byte[])**

24       method sends a data buffer to a resource using the method specified in the *method*  
25       parameter, and returns any response from the server. The URI of the resource to

1 receive the data. The method used to send the data to the resource. The data buffer  
2 to send to the resource.

3 **UploadFile**

4

5 [C#] public byte[] UploadFile(string address, string fileName);  
6 [C++] public: unsigned char UploadFile(String\* address, String\* fileName)  
7 \_\_gc[];  
8 [VB] Public Function UploadFile(ByVal address As String, ByVal fileName As  
9 String) As Byte()  
10 [JScript] public function UploadFile(address : String, fileName : String) : Byte[];  
11 Uploads a local file to a resource identified by a URI.

12

13 *Description*

14     Uploads the specified local file to the resource identified by a URI.

15 *Return Value:* An array of bytes containing the body of any response from the  
16 resource.

17     The **System.Net.WebClient.UploadFile(System.String, System.String)**  
18 method sends a local file to a resource. The underlying request is made using the  
19 POST method. The URI of the resource to receive the file. The file to send to the  
20 resource.

21     **UploadFile**

22

23 [C#] public byte[] UploadFile(string address, string method, string fileName);  
24 [C++] public: unsigned char UploadFile(String\* address, String\* method, String\*  
25 fileName) \_\_gc[];

```
1 [VB] Public Function UploadFile(ByVal address As String, ByVal method As
2 String, ByVal fileName As String) As Byte()
3
4 [JScript] public function UploadFile(address : String, method : String, fileName :
5 String) : Byte[];
```

#### 6 *Description*

7       Uploads the specified local file to the specified resource identified by a URI  
8 using the specified method.

9       *Return Value:* An array of bytes containing the body of any response from the  
10 resource.

11       The **System.Net.WebClient.UploadFile(System.String, System.String)**  
12 method sends a local file to a resource using the method specified in the *method*  
13 parameter, and returns any response from the server. The URI of the resource to  
14 receive the file. The method used to send the file to the resource. The file to send  
15 to the resource.

#### 16       UploadValues

17
18 [C#] public byte[] UploadValues(string address, NameValueCollection data);

19 [C++] public: unsigned char UploadValues(String\* address,  
20 NameValueCollection\* data) \_\_gc[];

21 [VB] Public Function UploadValues(ByVal address As String, ByVal data As  
22 NameValueCollection) As Byte()

23 [JScript] public function UploadValues(address : String, data :  
24 NameValueCollection) : Byte[]; Uploads a name/value collection to a resource  
25 identified by a URI.

1  
2 *Description*

3       Uploads the specified name/value collection to the specified resource  
4 identified by a URI.

5 *Return Value:* An array of bytes containing the body of any response from the  
6 resource.

7       The

8 **System.Net.WebClient.UploadValues(System.String, System.Collections.Speci**  
9 **alized.NameValueCollection)** method sends a  
10 **System.Collections.Specialized.NameValueCollection** to an Internet server. The  
11 underlying request is made using the POST method. The URI of the resource to  
12 receive the collection. The

13 **System.Collections.Specialized.NameValueCollection** to send to the resource.

14       UploadValues

15  
16 [C#] public byte[] UploadValues(string address, string method,  
17 NameValueCollection data);

18 [C++] public: unsigned char UploadValues(String\* address, String\* method,  
19 NameValueCollection\* data) \_\_gc[];

20 [VB] Public Function UploadValues(ByVal address As String, ByVal method As  
21 String, ByVal data As NameValueCollection) As Byte()

22 [JScript] public function UploadValues(address : String, method : String, data :  
23 NameValueCollection) : Byte[];

24  
25 *Description*

1       Uploads the specified name/value collection to the specified resource  
2       identified by a URI using the specified method.  
3       *Return Value:* An array of bytes containing the body of any response from the  
4       resource.  
5       The  
6       **System.Net.WebClient.UploadValues(System.String, System.Collections.Speci**  
7       **alized.NameValueCollection)** method sends a  
8       **System.Collections.Specialized.NameValueCollection** to a resource using the  
9       method specified in the *method* parameter, and returns any response from the  
10       server. The URI of the resource to receive the collection. The method used to send  
11       the file to the resource. The  
12       **System.Collections.Specialized.NameValueCollection** to send to the resource.

13       WebException class (System.Net)

14       UploadValues

17       *Description*

18       The exception that is thrown when an error occurs while accessing the  
19       network through a pluggable protocol.

20       The **System.Net.WebException** class is thrown by classes descended from  
21       **System.Net.WebRequest** and **System.Net.WebResponse** that implement  
22       pluggable protocols for accessing the Internet.

23       WebException

24       *Example Syntax:*

25       UploadValues

1  
2 [C#] public WebException();  
3 [C++] public: WebException();  
4 [VB] Public Sub New()  
5 [JScript] public function WebException(); Initializes a new instance of the  
6 **System.Net.WebException** class.  
7

8 *Description*

9     Initializes a new instance of the **System.Net.WebException** class.

10    The default constructor initializes a new instance of the

11 **System.Net.WebException** class with all fields set to **null** .

12    WebException

13    *Example Syntax:*

14    UploadValues

15  
16 [C#] public WebException(string message);

17 [C++] public: WebException(String\* message);

18 [VB] Public Sub New(ByVal message As String)

19 [JScript] public function WebException(message : String);

21 *Description*

22    Initializes a new instance of the **System.Net.WebException** class with the  
23 specified error message.

1        The **System.Net.WebException** instance is initialized with the  
2        **System.Exception.Message** property set to the value of *message* . The text of the  
3        error message.

4        WebException

5        *Example Syntax:*

6        UploadValues

7  
8        [C#] protected WebException(SerializationInfo serializationInfo,

9        StreamingContext streamingContext);

10        [C++] protected: WebException(SerializationInfo\* serializationInfo,

11        StreamingContext streamingContext);

12        [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal

13        streamingContext As StreamingContext)

14        [JScript] protected function WebException(serializationInfo : SerializationInfo,

15        streamingContext : StreamingContext);

16        WebException

17        *Example Syntax:*

18        UploadValues

19  
20        [C#] public WebException(string message, Exception innerException);

21        [C++] public: WebException(String\* message, Exception\* innerException);

22        [VB] Public Sub New(ByVal message As String, ByVal innerException As

23        Exception)

24        [JScript] public function WebException(message : String, innerException :

25        Exception);

1  
2 *Description*

3       Initializes a new instance of the **System.Net.WebException** class with the  
4       specified error message and nested exception.

5       The **System.Net.WebException** instance is initialized with the  
6       **System.Exception.Message** property set to the value of *message* and the  
7       **System.Exception.InnerException** property set to the value of *innerException* .  
8       The text of the error message. A nested exception.

9       WebException

10      *Example Syntax:*

11      UploadValues

12  
13 [C#] public WebException(string message, WebExceptionStatus status);  
14 [C++] public: WebException(String\* message, WebExceptionStatus status);  
15 [VB] Public Sub New(ByVal message As String, ByVal status As  
16 WebExceptionStatus)  
17 [JScript] public function WebException(message : String, status :  
18 WebExceptionStatus);

19  
20 *Description*

21       Initializes a new instance of the **System.Net.WebException** class with the  
22       specified error message and status.

23       The **System.Net.WebException** instance is initialized with the  
24       **System.Exception.Message** property set to the value of *message* and the

1      **System.Net.WebException.Status** property set to the value of *status* . The text of  
2      the error message. One of the **System.Net.WebExceptionStatus** values.

3      **WebException**

4      *Example Syntax:*

5      **UploadValues**

6

7      [C#] public WebException(string message, Exception innerException,  
8      WebExceptionStatus status, WebResponse response);

9      [C++] public: WebException(String\* message, Exception\* innerException,  
10     WebExceptionStatus status, WebResponse\* response);

11     [VB] Public Sub New(ByVal message As String, ByVal innerException As  
12     Exception, ByVal status As WebExceptionStatus, ByVal response As  
13     WebResponse)

14     [JScript] public function WebException(message : String, innerException :  
15     Exception, status : WebExceptionStatus, response : WebResponse);

16

17     *Description*

18         Initializes a new instance of the **System.Net.WebException** class with the  
19         specified error message, nested exception, status, and response.

20         The **System.Net.WebException** instance is initialized with the  
21         **System.Exception.Message** property set to the value of *message* , the  
22         **System.Exception.InnerException** property set to the value of *innerException* ,  
23         the **System.Net.WebException.Status** property set to the value of *status* , and the  
24         **System.Net.WebException.Response** property set to *response* . The text of the  
25         error message. A nested exception. One of the **System.Net.WebExceptionStatus**

1 values. A **System.Net.WebResponse** instance containing the response from the  
2 remote host.

3 **HelpLink**

4 **HResult**

5 **InnerException**

6 **Message**

7 **Response**

8 **UploadValues**

9

10

11 *Description*

12 Gets the response that the remote host returned.

13 Some Internet protocols, such as HTTP, return otherwise valid responses  
14 indicating that an error has occurred at the protocol level. When the response to an  
15 Internet request indicates an error, **System.Net.WebRequest.GetResponse** sets  
16 the **System.Net.WebException.Status** property to  
17 **System.Net.WebExceptionStatus.ProtocolError** and provides the  
18 **System.Net.WebResponse** containing the error message in the  
19 **System.Net.WebException.Response** property of the  
20 **System.Net.WebException** that was thrown. The application can examine the  
21 **System.Net.WebResponse** to determine the actual error.

22 **Source**

23 **StackTrace**

24 **Status**

25 **UploadValues**

1  
2  
3 *Description*

4 Gets the status of the response.

5 The **System.Net.WebException.Status** property indicates the reason for  
6 the **System.Net.WebException**.

7 TargetSite

8 ISerializable.GetObjectData

9  
10 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,  
11 StreamingContext streamingContext);

12 [C++] void ISerializable::GetObjectData(SerializationInfo\* serializationInfo,  
13 StreamingContext streamingContext);

14 [VB] Sub GetObjectData( ByVal serializationInfo As SerializationInfo, ByVal  
15 streamingContext As StreamingContext) Implements ISerializable.GetObjectData

16 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,  
17 streamingContext : StreamingContext);

18 WebExceptionStatus enumeration (System.Net)

19 ToString

20  
21  
22 *Description*

23 Defines status codes for the **System.Net.WebException** class.

24 The **System.Net.WebExceptionStatus** enumeration defines the status  
25 codes assigned to the **System.Net.WebException.Status** property.

```
1    ToString  
2  
3 [C#] public const WebExceptionStatus ConnectFailure;  
4 [C++] public: const WebExceptionStatus ConnectFailure;  
5 [VB] Public Const ConnectFailure As WebExceptionStatus  
6 [JScript] public var ConnectFailure : WebExceptionStatus;  
7  
8 Description
```

The remote service point could not be contacted at the transport level.

```
9    ToString  
10  
11  
12 [C#] public const WebExceptionStatus ConnectionClosed;  
13 [C++] public: const WebExceptionStatus ConnectionClosed;  
14 [VB] Public Const ConnectionClosed As WebExceptionStatus  
15 [JScript] public var ConnectionClosed : WebExceptionStatus;  
16  
17 Description
```

The connection was prematurely closed.

```
18    ToString  
19  
20  
21 [C#] public const WebExceptionStatus KeepAliveFailure;  
22 [C++] public: const WebExceptionStatus KeepAliveFailure;  
23 [VB] Public Const KeepAliveFailure As WebExceptionStatus  
24 [JScript] public var KeepAliveFailure : WebExceptionStatus;  
25
```

1  
2 *Description*

3       The connection for a request that specifies the Keep-alive header was  
4       closed unexpectedly.

5       *ToString*

6  
7 [C#] public const WebExceptionStatus NameResolutionFailure;  
8 [C++] public: const WebExceptionStatus NameResolutionFailure;  
9 [VB] Public Const NameResolutionFailure As WebExceptionStatus  
10 [JScript] public var NameResolutionFailure : WebExceptionStatus;

11  
12 *Description*

13       The name resolver service could not resolve the host name.

14       *ToString*

15  
16 [C#] public const WebExceptionStatus Pending;  
17 [C++] public: const WebExceptionStatus Pending;  
18 [VB] Public Const Pending As WebExceptionStatus  
19 [JScript] public var Pending : WebExceptionStatus;

20  
21 *Description*

22       An internal asynchronous request is pending.

23       *ToString*

24  
25 [C#] public const WebExceptionStatus PipelineFailure;

```
1 [C++] public: const WebExceptionStatus PipelineFailure;  
2 [VB] Public Const PipelineFailure As WebExceptionStatus  
3 [JScript] public var PipelineFailure : WebExceptionStatus;
```

4  
5 *Description*

6     ToString

```
7  
8 [C#] public const WebExceptionStatus ProtocolError;  
9 [C++] public: const WebExceptionStatus ProtocolError;  
10 [VB] Public Const ProtocolError As WebExceptionStatus  
11 [JScript] public var ProtocolError : WebExceptionStatus;
```

12  
13 *Description*

14     The response received from the server was complete but indicated a  
15     protocol-level error. For example, an HTTP protocol error such as 401 Access  
16     Denied would use this status.

17     ToString

```
18  
19 [C#] public const WebExceptionStatus ProxyNameResolutionFailure;  
20 [C++] public: const WebExceptionStatus ProxyNameResolutionFailure;  
21 [VB] Public Const ProxyNameResolutionFailure As WebExceptionStatus  
22 [JScript] public var ProxyNameResolutionFailure : WebExceptionStatus;
```

23  
24 *Description*

25     The name resolver service could not resolve the proxy host name.

1       ToString  
2  
3       [C#] public const WebExceptionStatus ReceiveFailure;  
4       [C++] public: const WebExceptionStatus ReceiveFailure;  
5       [VB] Public Const ReceiveFailure As WebExceptionStatus  
6       [JScript] public var ReceiveFailure : WebExceptionStatus;  
7

8       *Description*

9       A complete response was not received from the remote server.

10      ToString

11  
12     [C#] public const WebExceptionStatus RequestCanceled;  
13     [C++] public: const WebExceptionStatus RequestCanceled;  
14     [VB] Public Const RequestCanceled As WebExceptionStatus  
15     [JScript] public var RequestCanceled : WebExceptionStatus;  
16

17      *Description*

18      The request was canceled or the **System.Net.WebRequest.Abort** method  
19     was called.

20      ToString

21  
22     [C#] public const WebExceptionStatus SecureChannelFailure;  
23     [C++] public: const WebExceptionStatus SecureChannelFailure;  
24     [VB] Public Const SecureChannelFailure As WebExceptionStatus  
25     [JScript] public var SecureChannelFailure : WebExceptionStatus;

1  
2 *Description*

3 An error occurred in a secure channel link.

4 *ToString*

5  
6 [C#] public const WebExceptionStatus SendFailure;  
7 [C++] public: const WebExceptionStatus SendFailure;  
8 [VB] Public Const SendFailure As WebExceptionStatus  
9 [JScript] public var SendFailure : WebExceptionStatus;

10  
11 *Description*

12 A complete request could not be sent to the remote server.

13 *ToString*

14  
15 [C#] public const WebExceptionStatus ServerProtocolViolation;  
16 [C++] public: const WebExceptionStatus ServerProtocolViolation;  
17 [VB] Public Const ServerProtocolViolation As WebExceptionStatus  
18 [JScript] public var ServerProtocolViolation : WebExceptionStatus;

19  
20 *Description*

21 The server response was not a valid HTTP response.

22 *ToString*

23  
24 [C#] public const WebExceptionStatus Success;  
25 [C++] public: const WebExceptionStatus Success;

1 [VB] Public Const Success As WebExceptionStatus  
2 [JScript] public var Success : WebExceptionStatus;  
3  
4 *Description*  
5 No error was encountered. This is the default value for  
6 **System.Net.WebException.Status** .

7 ToString  
8  
9 [C#] public const WebExceptionStatus Timeout;  
10 [C++] public: const WebExceptionStatus Timeout;  
11 [VB] Public Const Timeout As WebExceptionStatus  
12 [JScript] public var Timeout : WebExceptionStatus;  
13

14 *Description*  
15 No response was received during the timeout period for a request.  
16 ToString  
17

18 [C#] public const WebExceptionStatus TrustFailure;  
19 [C++] public: const WebExceptionStatus TrustFailure;  
20 [VB] Public Const TrustFailure As WebExceptionStatus  
21 [JScript] public var TrustFailure : WebExceptionStatus;  
22

23 *Description*  
24 A server certificate could not be validated.  
25 WebHeaderCollection class (System.Net)

1       ToString

2

3

4       *Description*

5       Contains protocol headers associated with a request or response.

6       The **System.Net.WebHeaderCollection** class is generally accessed

7       through **System.Net.WebRequest.Headers** or

8       **System.Net.WebResponse.Headers**. Some common headers are considered

9       restricted and are either exposed directly by the API (such as **Content-Type**) or

10       protected by the system and cannot be changed.

11       WebHeaderCollection

12       *Example Syntax:*

13       ToString

14

15       [C#] public WebHeaderCollection();

16       [C++] public: WebHeaderCollection();

17       [VB] Public Sub New()

18       [JScript] public function WebHeaderCollection(); Initializes a new instance of the

19       **System.Net.WebHeaderCollection** class.

20

21       *Description*

22       Initializes a new instance of the **System.Net.WebHeaderCollection** class.

23       WebHeaderCollection

24       *Example Syntax:*

25       ToString

1  
2 [C#] protected WebHeaderCollection(SerializationInfo serializationInfo,  
3 StreamingContext streamingContext);  
4 [C++] protected: WebHeaderCollection(SerializationInfo\* serializationInfo,  
5 StreamingContext streamingContext);  
6 [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal  
7 streamingContext As StreamingContext)  
8 [JScript] protected function WebHeaderCollection(serializationInfo :  
9 SerializationInfo, streamingContext : StreamingContext);

10  
11 *Description*

12       Initializes a new instance of the **System.Net.WebHeaderCollection** class  
13      from the specified instances of the  
14      **System.Runtime.Serialization.SerializationInfo** and  
15      **System.Runtime.Serialization.StreamingContext** classes.

16       This constructor implements the  
17      **System.Runtime.Serialization.ISerializable** interface for the  
18      **System.Net.WebHeaderCollection** class. A  
19      **System.Runtime.Serialization.SerializationInfo** containing the information  
20      required to serialize the **System.Net.WebHeaderCollection**. A  
21      **System.Runtime.Serialization.StreamingContext** containing the source of the  
22      serialized stream associated with the new **System.Net.WebHeaderCollection**.

23      AllKeys

24      Count

25      IsReadOnly

1           Item  
2           Item  
3           Keys  
4           Add  
5  
6 [C#] public void Add(string header);  
7 [C++] public: void Add(String\* header);  
8 [VB] Public Sub Add(ByVal header As String)  
9 [JScript] public function Add(header : String);

10  
11 *Description*

12           Inserts the specified header into the collection.  
13           The  
14 **System.Net.WebHeaderCollection.Add(System.String, System.String)** method  
15 inserts a new header into the list of header name/value pairs. The header value  
16 must be specified in the format "header:value". The header to add, with the name  
17 and value separated by a colon.

18           Add  
19  
20 [C#] public override void Add(string name, string value);  
21 [C++] public: void Add(String\* name, String\* value);  
22 [VB] Overrides Public Sub Add(ByVal name As String, ByVal value As String)  
23 [JScript] public override function Add(name : String, value : String); Inserts a new  
24 header into the collection.  
25

1  
2 *Description*

3       Inserts a new header with the specified name and value into the collection.

4       The

5       **System.Net.WebHeaderCollection.Add(System.String, System.String)** method  
6       inserts a new header into the list of header name/value pairs. The header to add to  
7       the collection. The content of the header.

8       AddWithoutValidate

9  
10      [C#] protected void AddWithoutValidate(string headerName, string headerValue);  
11      [C++] protected: void AddWithoutValidate(String\* headerName, String\*  
12       headerValue);  
13      [VB] Protected Sub AddWithoutValidate(ByName headerName As String, ByVal  
14       headerValue As String)  
15      [JScript] protected function AddWithoutValidate(headerName : String,  
16       headerValue : String);

17  
18 *Description*

19       Inserts a header into the collection without checking whether the header is  
20       on the restricted header list.

21       The

22       **System.Net.WebHeaderCollection.AddWithoutValidate(System.String, Syste**  
23       **m.String)** method adds a header to the collection without checking whether the  
24       header is on the restricted header list. The header to add to the collection. The  
25       content of the header.

1       GetValues  
2  
3       [C#] public override string[] GetValues(string header);  
4       [C++] public: String\* GetValues(String\* header) \_\_gc[];  
5       [VB] Overrides Public Function GetValues(ByVal header As String) As String()  
6       [JScript] public override function GetValues(header : String) : String[]; Gets an  
7       array of header values stored in a header.

8  
9       *Description*

10       Gets an array of header values stored in a header.

11       *Return Value:* An array of header strings.

12       **System.Net.WebHeaderCollection.GetValues(System.String)** returns the  
13       contents of the specified header as an array. The header to return.

14       IsRestricted  
15  
16

16       [C#] public static bool IsRestricted(string headerName);  
17       [C++] public: static bool IsRestricted(String\* headerName);  
18       [VB] Public Shared Function IsRestricted(ByVal headerName As String) As  
19       Boolean

20       [JScript] public static function IsRestricted(headerName : String) : Boolean;

21  
22       *Description*

23       Tests whether the specified HTTP header can be set.

24       *Return Value:* **true** if the header is restricted; otherwise **false** .

1       **The System.Net.WebHeaderCollection.IsRestricted(System.String)**

2       method returns **true** to indicate that a header is restricted and must be set using  
3       properties instead of directly. The restricted headers are: Accept Connection Date  
4       Expect Host Range Referer Tests if access to the HTTP header with the provided  
5       name is accessible for setting. The header to test.

6       **OnDeserialization**

7  
8       [C#] public override void OnDeserialization(object sender);  
9       [C++] public: void OnDeserialization(Object\* sender);  
10       [VB] Overrides Public Sub OnDeserialization(ByVal sender As Object)  
11       [JScript] public override function OnDeserialization(sender : Object);

12  
13       *Description*

14  
15       **Remove**

16  
17       [C#] public override void Remove(string name);  
18       [C++] public: void Remove(String\* name);  
19       [VB] Overrides Public Sub Remove(ByVal name As String)  
20       [JScript] public override function Remove(name : String);

21  
22       *Description*

23       Removes the specified header from the collection.

24       **System.Net.WebHeaderCollection.Remove(System.String)** deletes the  
25       specified header from the collection. If the same header was added multiple times

1 with **System.Net.WebHeaderCollection.Add(System.String, System.String)** , a  
2 single call to **System.Net.WebHeaderCollection.Remove(System.String)**  
3 deletes all of the headers. The name of the header to remove from the collection.

4 Set

5  
6 [C#] public override void Set(string name, string value);  
7 [C++] public: void Set(String\* name, String\* value);  
8 [VB] Overrides Public Sub Set(ByVal name As String, ByVal value As String)  
9 [JScript] public override function Set(name : String, value : String);

10  
11 *Description*

12 Sets the specified header to the specified value.

13 The

14 **System.Net.WebHeaderCollection.Set(System.String, System.String)** method  
15 inserts a new header into the list of header name/value pairs. The header value  
16 must be specified in the format header:value. The header to set. The content of the  
17 header to set.

18 **ISerializable.GetObjectData**

19  
20 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,  
21 StreamingContext streamingContext);  
22 [C++] void ISerializable::GetObjectData(SerializationInfo\* serializationInfo,  
23 StreamingContext streamingContext);  
24 [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal  
25 streamingContext As StreamingContext) Implements ISerializable.GetObjectData

1 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,  
2 streamingContext : StreamingContext);  
3  
4  
5 [C#] public byte[] ToByteArray();  
6 [C++] public: unsigned char ToByteArray() \_\_gc[];  
7 [VB] Public Function ToByteArray() As Byte()  
8 [JScript] public function ToByteArray() : Byte[];

9  
10 *Description*

11      Obsolete.

12      *ToString*

13  
14 [C#] public override string ToString();  
15 [C++] public: String\* ToString();  
16 [VB] Overrides Public Function ToString() As String  
17 [JScript] public override function ToString() : String;

18  
19 *Description*

20      Obsolete.

21      WebPermission class (System.Net)

22      *ToString*

23  
24  
25 *Description*

1       Controls rights to access an Internet resource.

2       WebPermission

3       *Example Syntax:*

4       ToString

6       [C#] public WebPermission();

7       [C++] public: WebPermission();

8       [VB] Public Sub New()

9       [JScript] public function WebPermission();

11      *Description*

12      Creates a new instance of the **System.Net.WebPermission** class.

13      WebPermission

14      *Example Syntax:*

15      ToString

17       [C#] public WebPermission(PermissionState state);

18       [C++] public: WebPermission(PermissionState state);

19       [VB] Public Sub New(ByVal state As PermissionState)

20       [JScript] public function WebPermission(state : PermissionState); Creates a new  
21       instance of the **System.Net.WebPermission** class.

23      *Description*

24      Creates a new instance of the **System.Net.WebPermission** class that  
25      passes all demands or fails all demands.

1        If **System.Net.WebPermission** is created with the **Unrestricted** value  
2 from **System.Security.Permissions.PermissionState** then the  
3 **System.Net.WebPermission** instance will pass all demands. Any other value for s  
4 tate will result in a **System.Net.WebPermission** instance that will fail all  
5 demands. One of the **System.Security.Permissions.PermissionState** values.

6        **WebPermission**

7        *Example Syntax:*

8        **ToString**

9  
10      [C#] public WebPermission(NetworkAccess access, Regex uriRegex);  
11      [C++] public: WebPermission(NetworkAccess access, Regex\* uriRegex);  
12      [VB] Public Sub New(ByVal access As NetworkAccess, ByVal uriRegex As  
13                    Regex)  
14      [JScript] public function WebPermission(access : NetworkAccess, uriRegex :  
15                    Regex);  
16

17        *Description*

18        Initializes a new instance of the **System.Net.WebPermission** class with the  
19 specified access rights for the specified URI regular expression. The access right  
20 to grant. A regular expression describing the URI to grant the access right to.

21        **WebPermission**

22        *Example Syntax:*

23        **ToString**

24  
25      [C#] public WebPermission(NetworkAccess access, string uriString);

```
1 [C++] public: WebPermission(NetworkAccess access, String* uriString);  
2 [VB] Public Sub New(ByVal access As NetworkAccess, ByVal uriString As  
3 String)  
4 [JScript] public function WebPermission(access : NetworkAccess, uriString :  
5 String);  
6  
7 Description
```

8       Initializes a new instance of the **System.Net.WebPermission** class with the  
9       specified access rights for the specified URI. The access right to grant. A regular  
10      expression describing the URI to grant the access right to.

```
11      AcceptList
```

```
12      ToString
```

```
14 [C#] public IEnumerator AcceptList {get;}
```

```
15 [C++] public: __property IEnumerator* get_AcceptList();
```

```
16 [VB] Public ReadOnly Property AcceptList As IEnumerator
```

```
17 [JScript] public function get AcceptList() : IEnumerator;
```

```
19 Description
```

20       Returns the enumeration of permissions to export a local URI.

```
21      ConnectList
```

```
22      ToString
```

```
24 [C#] public IEnumerator ConnectList {get;}
```

```
25 [C++] public: __property IEnumerator* get_ConnectList();
```

1 [VB] Public ReadOnly Property ConnectList As IEnumator  
2 [JScript] public function get ConnectList() : IEnumator;

3  
4 *Description*

5 Returns the enumeration of permissions to connect a remote URI.

6 AddPermission

7  
8 [C#] public void AddPermission(NetworkAccess access, Regex uriRegex);  
9 [C++] public: void AddPermission(NetworkAccess access, Regex\* uriRegex);  
10 [VB] Public Sub AddPermission( ByVal access As NetworkAccess, ByVal  
11 uriRegex As Regex)  
12 [JScript] public function AddPermission(access : NetworkAccess, uriRegex :  
13 Regex);

14  
15 *Description*

16 Adds a new instance of the **System.Net.WebPermission** class with the  
17 specified access rights for the specified URI pattern. The access right to grant. A  
18 regular expression describing the URI to grant the access right to.

19 AddPermission

20  
21 [C#] public void AddPermission(NetworkAccess access, string uriString);  
22 [C++] public: void AddPermission(NetworkAccess access, String\* uriString);  
23 [VB] Public Sub AddPermission( ByVal access As NetworkAccess, ByVal  
24 uriString As String)  
25 [JScript] public function AddPermission(access : NetworkAccess, uriString :

1     String); Adds a new instance of the **System.Net.WebPermission** class with the  
2     specified access rights for the specified URI.

3

4     *Description*

5         Adds a new instance of the **System.Net.WebPermission** class with the  
6         specified access rights for the specified URI. The access right to grant. A string  
7         describing the URI to grant the access right to.

8         *Copy*

9

10        [C#] public override IPermission Copy();  
11        [C++] public: IPermission\* Copy();  
12        [VB] Overrides Public Function Copy() As IPermission  
13        [JScript] public override function Copy() : IPermission;

14

15     *Description*

16         Creates a copy of a **System.Net.WebPermission** instance.

17     *Return Value:* A new instance of the **System.Net.WebPermission** class that is a  
18     copy of the current instance.

19         *FromXml*

20

21        [C#] public override void FromXml(SecurityElement securityElement);  
22        [C++] public: void FromXml(SecurityElement\* securityElement);  
23        [VB] Overrides Public Sub FromXml(ByVal securityElement As  
24                    SecurityElement)  
25        [JScript] public override function FromXml(securityElement : SecurityElement);

1           *Description*

2           Reconstructs a **System.Net.WebPermission** instance from an XML  
3           encoding.

4           The

5           **System.Net.WebPermission.FromXml(System.Security.SecurityElement)**

6           method reconstructs a **System.Net.WebPermission** instance from an XML  
7           encoding defined by the **System.Security.SecurityElement** class. The XML  
8           encoding to use to reconstruct the **System.Net.WebPermission** instance.

9           Intersect

10           [C#] public override IPermission Intersect(IPermission target);

11           [C++] public: IPermission\* Intersect(IPPermission\* target);

12           [VB] Overrides Public Function Intersect(ByVal target As IPermission) As  
13           IPermission

14           [JScript] public override function Intersect(target : IPermission) : IPermission;

15           *Description*

16           Returns the logical intersection between two **System.Net.WebPermission**  
17           instances.

18           *Return Value:* **System.Net.WebPermission** instance that represents the  
19           intersection of two **System.Net.WebPermission** instances. The  
20           **System.Net.WebPermission** instance to combine with the current instance.

21           IsSubsetOf

1  
2 [C#] public override bool IsSubsetOf(IPermission target);  
3 [C++] public: bool IsSubsetOf(IPermission\* target);  
4 [VB] Overrides Public Function IsSubsetOf(ByVal target As IPermission) As  
5 Boolean  
6 [JScript] public override function IsSubsetOf(target : IPermission) : Boolean;

7  
8 *Description*

9     Compares two **System.Net.WebPermission** instances.

10    *Return Value:* **false** if *target* is **null** ; otherwise an exception is thrown.

11    The

12   **System.Net.WebPermission.IsSubsetOf(System.Security.IPermission)** method  
13   is not supported by the **System.Net.WebPermission** class. The second  
14   **System.Net.WebPermission** instance to compare.

15    **IsUnrestricted**

16  
17 [C#] public bool IsUnrestricted();

18 [C++] public: \_\_sealed bool IsUnrestricted();

19 [VB] NotOverridable Public Function IsUnrestricted() As Boolean

20 [JScript] public function IsUnrestricted() : Boolean;

21  
22 *Description*

23    Checks the overall permission state of the object.

24    *Return Value:* true if the **System.Net.WebPermission** instance was created with

1 the **Unrestricted** value from **System.Security.Permissions.PermissionState** ;  
2 otherwise, **false** .

3 **ToXml**

4  
5 [C#] public override SecurityElement ToXml();  
6 [C++] public: SecurityElement\* ToXml();  
7 [VB] Overrides Public Function ToXml() As SecurityElement  
8 [JScript] public override function ToXml() : SecurityElement;

9  
10 *Description*

11       Creates an XML encoding of a **System.Net.WebPermission** instance and  
12       its current state.

13       *Return Value:* A **System.Security.SecurityElement** instance containing an XML-  
14       encoded representation of the **System.Net.WebPermission** instance, including  
15       state information.

16       The **System.Net.WebPermission.ToXml** method creates a  
17       **System.Security.SecurityElement** instance to XML-encode a representation of  
18       the **System.Net.WebPermission** instance, including state information.

19       **Union**

20  
21 [C#] public override IPermission Union(IPermission target);  
22 [C++] public: IPermission\* Union(IPPermission\* target);  
23 [VB] Overrides Public Function Union(ByVal target As IPermission) As  
24 IPermission  
25 [JScript] public override function Union(target : IPermission) : IPermission;

1           *Description*

2           Returns the logical union between two **System.Net.WebPermission**  
3           instances.

4           *Return Value:* **System.Net.WebPermission** instance that represents the union of  
5           two **System.Net.WebPermission** instances. The **System.Net.WebPermission**  
6           instance to combine with the current instance.

7           WebPermissionAttribute class (System.Net)

8           Union

9

10

11           *Description*

12           Enables security actions for **System.Net.WebPermission** to be applied to  
13           code using declarative security. This class cannot be inherited.

14           WebPermissionAttribute

15           *Example Syntax:*

16           Union

17           [C#] public WebPermissionAttribute(SecurityAction action);

18

19           [C++] public: WebPermissionAttribute(SecurityAction action);

20

21           [VB] Public Sub New(ByVal action As SecurityAction)

22           [JScript] public function WebPermissionAttribute(action : SecurityAction);

23

24           *Description*

1       Initializes a new instance of the **System.Net.WebPermissionAttribute**  
2    class with the specified **System.Security.Permissions.SecurityAction** . One of  
3    the **System.Security.Permissions.SecurityAction** values.

4       Accept

5       Union

7       [C#] public string Accept {get; set;}

8       [C++] public: \_\_property String\* get\_Accept();public: \_\_property void  
9       set\_Accept(String\*);

10      [VB] Public Property Accept As String

11      [JScript] public function get Accept() : String;public function set Accept(String);

13      *Description*

14       Gets or sets the URI accepted by this permission instance.

15       AcceptPattern

16       Union

18       [C#] public string AcceptPattern {get; set;}

19       [C++] public: \_\_property String\* get\_AcceptPattern();public: \_\_property void  
20       set\_AcceptPattern(String\*);

21      [VB] Public Property AcceptPattern As String

22      [JScript] public function get AcceptPattern() : String;public function set  
23       AcceptPattern(String);

25      *Description*

1       Gets or sets a regular expression pattern that describes the URI accepted by  
2 this permission instance.

3       Action

4       Connect

5       Union

6

7

8 *Description*

9       Gets or sets the URI connection controlled by this permission instance.

10      ConnectPattern

11      Union

12

13 [C#] public string ConnectPattern {get; set;}

14 [C++] public: \_\_property String\* get\_ConnectPattern();public: \_\_property void  
15 set\_ConnectPattern(String\*);

16 [VB] Public Property ConnectPattern As String

17 [JScript] public function get ConnectPattern() : String;public function set  
18 ConnectPattern(String);

19

20 *Description*

21       Gets or sets a regular expression pattern that describes the URI connection  
22 controlled by this permission instance.

23      TypeId

24      Unrestricted

25      CreatePermission

1  
2 [C#] public override IPermission CreatePermission();  
3 [C++] public: IPermission\* CreatePermission();  
4 [VB] Overrides Public Function CreatePermission() As IPermission  
5 [JScript] public override function CreatePermission() : IPermission;

6  
7 *Description*

8       Creates and returns a new instance of the **System.Net.WebPermission**  
9       class.

10      *Return Value:* A **System.Net.WebPermission** instance corresponding to the  
11        security declaration.

12        WebProxy class (System.Net)  
13        ToString

14  
15  
16 *Description*

17       Contains HTTP proxy settings for the **System.Net.WebRequest** class.  
18       The **System.Net.WebProxy** class contains the proxy settings that  
19       **System.Net.WebRequest** instances use to override the proxy settings in  
20       **System.Net.GlobalProxySelection** .

21       WebProxy

22       *Example Syntax:*

23       ToString

24  
25 [C#] public WebProxy();

```
1 [C++] public: WebProxy();  
2 [VB] Public Sub New()  
3 [JScript] public function WebProxy(); Initializes a new instance of the  
4 System.Net.WebProxy class.  
5
```

#### 6 *Description*

7     Initializes an empty instance of the **System.Net.WebProxy** class.  
8     The default constructor initializes an empty instance of the  
9     **System.Net.WebProxy** class initializes an empty instance of the  
10  **System.Net.WebProxy** class with the **System.Net.WebProxy.Address** property  
11  set to **null**. When the **System.Net.WebProxy.Address** property is **null** , the  
12  **System.Net.WebProxy.IsBypassed(System.Uri)** method returns **true** , and the  
13  **System.Net.WebProxy.GetProxy(System.Uri)** method returns the destination  
14  address.

15     WebProxy

16     *Example Syntax:*

17     ToString

```
18  
19 [C#] public WebProxy(string Address);  
20 [C++] public: WebProxy(String* Address);  
21 [VB] Public Sub New(ByVal Address As String)  
22 [JScript] public function WebProxy(Address : String);  
23
```

#### 24 *Description*

1       Initializes a new instance of the **System.Net.WebProxy** class with the  
2       specified URI.

3       The **System.Net.WebProxy** instance is initialized with the  
4       **System.Net.WebProxy.Address** property set to a **System.Uri** instance containing  
5       *Address* . The URI of the proxy server.

6       WebProxy

7       *Example Syntax:*

8       ToString

9  
10      [C#] public WebProxy(Uri Address);  
11      [C++] public: WebProxy(Uri\* Address);  
12      [VB] Public Sub New(ByVal Address As Uri)  
13      [JScript] public function WebProxy(Address : Uri);

14  
15      *Description*

16       Initializes a new instance of the **System.Net.WebProxy** class from the  
17       specified **System.Uri** .

18       The **System.Net.WebProxy** instance is initialized with the  
19       **System.Net.WebProxy.Address** property set to the *Address* parameter. A  
20       **System.Uri** containing the address of the proxy server.

21       WebProxy

22       *Example Syntax:*

23       ToString

24  
25      [C#] protected WebProxy(SerializationInfo serializationInfo, StreamingContext

```
1  streamingContext);  
2  [C++] protected: WebProxy(SerializationInfo* serializationInfo,  
3  StreamingContext streamingContext);  
4  [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal  
5  streamingContext As StreamingContext)  
6  [JScript] protected function WebProxy(serializationInfo : SerializationInfo,  
7  streamingContext : StreamingContext);  
8  
9  Description
```

10       Initializes a new instance of the **System.Net.WebProxy** class from the  
11 specified instances of the **System.Runtime.Serialization.SerializationInfo** and  
12 **System.Runtime.Serialization.StreamingContext** classes.

13       This constructor implements the  
14 **System.Runtime.Serialization.ISerializable** interface for the  
15 **System.Net.WebProxy** class. A  
16 **System.Runtime.Serialization.SerializationInfo** containing the information  
17 required to serialize the new **System.Net.WebProxy** instance. A  
18 **System.Runtime.Serialization.StreamingContext** containing the source and  
19 destination of the serialized stream associated with the new  
20 **System.Net.WebProxy** .

21 WebProxy

22 *Example Syntax:*

23 ToString

25 [C#] public WebProxy(string Address, bool BypassOnLocal);

```
1 [C++] public: WebProxy(String* Address, bool BypassOnLocal);  
2 [VB] Public Sub New(ByVal Address As String, ByVal BypassOnLocal As  
3 Boolean)  
4 [JScript] public function WebProxy(Address : String, BypassOnLocal : Boolean);  
5
```

#### 6 *Description*

7       Initializes a new instance of the **System.Net.WebProxy** class with the  
8 specified URI and bypass setting.

9       The **System.Net.WebProxy** instance is initialized with the  
10 **System.Net.WebProxy.Address** property set to a **System.Uri** instance containing  
11 *Address* and the **System.Net.WebProxy.BypassProxyOnLocal** property set to  
12 *BypassOnLocal*. The URI of the proxy server. **true** to bypass the proxy for local  
13 addresses; otherwise, **false**.

14       WebProxy

15       *Example Syntax:*

16       ToString

```
17  
18 [C#] public WebProxy(string Host, int Port);  
19 [C++] public: WebProxy(String* Host, int Port);  
20 [VB] Public Sub New(ByVal Host As String, ByVal Port As Integer)  
21 [JScript] public function WebProxy(Host : String, Port : int);  
22
```

#### 23 *Description*

24       Initializes a new instance of the **System.Net.WebProxy** class with the  
25 specified host and port number.

1        The **System.Net.WebProxy** instance is initialized with the  
2        **System.Net.WebProxy.Address** property set to a **System.Uri** instance of the  
3        form *http:// Host : Port* . The name of the proxy host. The port number on *Host* to  
4        use.

5        **WebProxy**

6        *Example Syntax:*

7        **ToString**

8

9        [C#] public WebProxy(Uri Address, bool BypassOnLocal);  
10       [C++] public: WebProxy(Uri\* Address, bool BypassOnLocal);  
11       [VB] Public Sub New(ByVal Address As Uri, ByVal BypassOnLocal As  
12       Boolean)  
13       [JScript] public function WebProxy(Address : Uri, BypassOnLocal : Boolean);

14

15       *Description*

16       Initializes a new instance of the **System.Net.WebProxy** class with the  
17       **System.Uri** and bypass setting.

18       The **System.Net.WebProxy** instance is initialized with the  
19       **System.Net.WebProxy.Address** property set to *Address* and with the  
20       **System.Net.WebProxy.BypassProxyOnLocal** property set to *BypassOnLocal* . A  
21       **System.Uri** containing the address of the proxy server. **true** to bypass the proxy  
22       for local addresses; otherwise, **false**.

23       **WebProxy**

24       *Example Syntax:*

25       **ToString**

```
1  
2 [C#] public WebProxy(string Address, bool BypassOnLocal, string[] BypassList);  
3 [C++] public: WebProxy(String* Address, bool BypassOnLocal, String*  
4 BypassList __gc[]);  
5 [VB] Public Sub New(ByVal Address As String, ByVal BypassOnLocal As  
6 Boolean, ByVal BypassList() As String)  
7 [JScript] public function WebProxy(Address : String, BypassOnLocal : Boolean,  
8 BypassList : String[]);  
9  
10 Description
```

11       Initializes a new instance of the **System.Net.WebProxy** class with the  
12 specified URI, bypass setting, and list of URIs to bypass.

13       The **System.Net.WebProxy** instance is initialized with the  
14 **System.Net.WebProxy.Address** property set to a **System.Uri** instance containing  
15 *Address* , the **System.Net.WebProxy.BypassProxyOnLocal** property set to  
16 *BypassOnLocal* , and the **System.Net.WebProxy.BypassList** property set to  
17 *BypassList* . The URI of the proxy server. **true** to bypass the proxy for local  
18 addresses; otherwise, **false**. An array of regular expression strings containing the  
19 URIs of the servers to bypass.

20       WebProxy

21       *Example Syntax:*

22       ToString

```
23  
24 [C#] public WebProxy(Uri Address, bool BypassOnLocal, string[] BypassList);  
25 [C++] public: WebProxy(Uri* Address, bool BypassOnLocal, String* BypassList
```

```
1  __gc[]);  
2  [VB] Public Sub New(ByVal Address As Uri, ByVal BypassOnLocal As Boolean,  
3  ByVal BypassList() As String)  
4  [JScript] public function WebProxy(Address : Uri, BypassOnLocal : Boolean,  
5  BypassList : String[]);  
6  
7  Description
```

8       Initializes a new instance of the **System.Net.WebProxy** class with the  
9       specified **System.Uri** , bypass setting, and list of URIs to bypass.

10      The **System.Net.WebProxy** instance is initialized with the  
11     **System.Net.WebProxy.Address** property set to *Address* , the  
12     **System.Net.WebProxy.BypassProxyOnLocal** property set to *BypassOnLocal* ,  
13     and the **System.Net.WebProxy.BypassList** property set to *BypassList* . A  
14     **System.Uri** containing the address of the proxy server. **true** to bypass the proxy  
15     for local addresses; otherwise, **false**. An array of regular expression strings  
16     containing the URIs of the servers to bypass.

17      WebProxy

18      *Example Syntax:*

19      ToString

```
20  
21 [C#] public WebProxy(string Address, bool BypassOnLocal, string[] BypassList,  
22 ICredentials Credentials);  
23 [C++] public: WebProxy(String* Address, bool BypassOnLocal, String*  
24 BypassList __gc[], ICredentials* Credentials);  
25 [VB] Public Sub New(ByVal Address As String, ByVal BypassOnLocal As
```

```
1 Boolean, ByVal BypassList() As String, ByVal Credentials As ICredentials)
2 [JScript] public function WebProxy(Address : String, BypassOnLocal : Boolean,
3 BypassList : String[], Credentials : ICredentials);
4
```

```
5 Description
```

```
6     Initializes a new instance of the System.Net.WebProxy class with the
7     specified URI, bypass setting, list of URIs to bypass, and credentials.
```

```
8     The System.Net.WebProxy instance is initialized with the
9     System.Net.WebProxy.Address property set to a System.Uri instance containing
10    Address , the System.Net.WebProxy.BypassProxyOnLocal property set to
11    BypassOnLocal , the System.Net.WebProxy.BypassList property set to
12    BypassList , and the System.Net.WebProxy.Credentials property set to
13    Credentials . The URI of the proxy server. true to bypass the proxy for local
14    addresses; otherwise, false. An array of regular expression strings containing the
15    URIs of the servers to bypass. An System.Net.ICredentials to submit to the proxy
16    server for authentication.
```

```
17     WebProxy
```

```
18     Example Syntax:
```

```
19     ToString
```

```
20
21 [C#] public WebProxy(Uri Address, bool BypassOnLocal, string[] BypassList,
22 ICredentials Credentials);
23 [C++] public: WebProxy(Uri* Address, bool BypassOnLocal, String* BypassList
24     __gc[], ICredentials* Credentials);
25 [VB] Public Sub New(ByVal Address As Uri, ByVal BypassOnLocal As Boolean,
```

```
1  ByVal BypassList() As String, ByVal Credentials As ICredentials)
2  [JScript] public function WebProxy(Address : Uri, BypassOnLocal : Boolean,
3  BypassList : String[], Credentials : ICredentials);
4
```

```
5  Description
```

```
6      Initializes a new instance of the System.Net.WebProxy class with the
7      specified System.Uri , bypass setting, list of URIs to bypass, and credentials.
8
9      The System.Net.WebProxy instance is initialized with the
10     System.Net.WebProxy.Address property set to Address , the
11     System.Net.WebProxy.BypassProxyOnLocal property set to BypassOnLocal ,
12     the System.Net.WebProxy.BypassList property set to BypassList , and the
13     System.Net.WebProxy.Credentials property set to Credentials A System.Uri
14     containing the address of the proxy server. true to bypass the proxy for local
15     addresses; otherwise, false. An array of regular expression strings containing the
16     URIs of the servers to bypass. An System.Net.ICredentials to submit to the proxy
17     server for authentication.
```

```
18     Address
```

```
19     ToString
```

```
20 [C#] public Uri Address {get; set;}
```

```
21 [C++] public: __property Uri* get_Address();public: __property void
22 set_Address(Uri*);
```

```
23 [VB] Public Property Address As Uri
```

```
24 [JScript] public function get Address() : Uri;public function set Address(Uri);
```

1  
2 *Description*  
3     Gets or sets the address of the proxy server.  
4     The **System.Net.WebProxy.Address** property contains the address of the  
5 proxy server. When **System.Net.WebProxy.Address** is **null** , all requests bypass  
6 the proxy and connect directly to the destination host.

7     BypassArrayList  
8     ToString

9  
10 [C#] public ArrayList BypassArrayList {get;}  
11 [C++] public: \_\_property ArrayList\* get\_BypassArrayList();  
12 [VB] Public ReadOnly Property BypassArrayList As ArrayList  
13 [JScript] public function get BypassArrayList() : ArrayList;

14  
15 *Description*

16     Gets a list of addresses that do not use the proxy server.  
17     The **System.Net.WebProxy.BypassList** property contains an array of URI  
18 strings that a **System.Net.WebRequest** instance accesses directly instead of  
19 through the proxy server.

20     BypassList  
21     ToString

22  
23 [C#] public string[] BypassList {get; set;}  
24 [C++] public: \_\_property String\* get\_BypassList();public: \_\_property void  
25 set\_BypassList(String\* \_\_gc[]);

```
1 [VB] Public Property BypassList As String ()  
2 [JScript] public function get BypassList() : String[];public function set  
3 BypassList(String[]);  
4
```

```
5 Description
```

```
6 Gets or sets an array of addresses that do not use the proxy server.
```

```
7 The System.Net.WebProxy.BypassList property contains an array of  
8 regular expressions that describe URIs that a System.Net.WebRequest instance  
9 accesses directly instead of through the proxy server.
```

```
10 BypassProxyOnLocal
```

```
11 ToString
```

```
12  
13 [C#] public bool BypassProxyOnLocal {get; set;}
```

```
14 [C++] public: __property bool get_BypassProxyOnLocal();public: __property  
15 void set_BypassProxyOnLocal(bool);
```

```
16 [VB] Public Property BypassProxyOnLocal As Boolean
```

```
17 [JScript] public function get BypassProxyOnLocal() : Boolean;public function set  
18 BypassProxyOnLocal(Boolean);
```

```
19  
20 Description
```

```
21 Gets or sets a value indicating whether to bypass the proxy server for local  
22 addresses.
```

```
23 The setting of the System.Net.WebProxy.BypassProxyOnLocal property  
24 determines whether System.Net.WebRequest instances use the proxy server  
25 when accessing local Internet resources.
```

1            Credentials  
2            ToString  
3  
4 [C#] public ICredentials Credentials {get; set;}  
5 [C++] public: \_\_property ICredentials\* get\_Credentials();public: \_\_property void  
6 set\_Credentials(ICredentials\*);  
7 [VB] Public Property Credentials As ICredentials  
8 [JScript] public function get Credentials() : ICredentials;public function set  
9 Credentials(ICredentials);

10  
11 *Description*

12            Gets or sets the credentials to submit to the proxy server for authentication.  
13            The **System.Net.WebProxy.Credentials** property contains the  
14 authentication credentials to send to the proxy server in response to an HTTP 407  
15 (proxy authorization) status code.

16            GetDefaultProxy  
17  
18 [C#] public static WebProxy GetDefaultProxy();  
19 [C++] public: static WebProxy\* GetDefaultProxy();  
20 [VB] Public Shared Function GetDefaultProxy() As WebProxy  
21 [JScript] public static function GetDefaultProxy() : WebProxy;

22  
23 *Description*  
24  
25

1       Reads the Internet Explorer nondynamic proxy settings.

2       *Return Value:* A **System.Net.WebProxy** instance containing the nondynamic  
3       proxy settings from Internet Explorer 5.5.

4       The **System.Net.WebProxy.GetDefaultProxy** method reads the  
5       nondynamic proxy settings stored by Internet Explorer 5.5 and creates a  
6       **System.Net.WebProxy** instance with those settings.

7       **GetProxy**

8

9       [C#] public Uri GetProxy(Uri destination);  
10      [C++] public: \_\_sealed Uri\* GetProxy(Uri\* destination);  
11      [VB] NotOverridable Public Function GetProxy( ByVal destination As Uri) As Uri  
12      [JScript] public function GetProxy(destination : Uri) : Uri;

13

14       **Description**

15       Returns the proxied URI for a request.

16       *Return Value:* The **System.Uri** of the Internet resource, if the resource is on the  
17       bypass list; otherwise, the **System.Uri** of the proxy.

18       The **System.Net.WebProxy.GetProxy(System.Uri)** method returns the  
19       URI that the **System.Net.WebRequest** uses to access the Internet resource. The  
20       **System.Uri** of the requested Internet resource.

21       **IsBypassed**

22

23      [C#] public bool IsBypassed(Uri host);  
24      [C++] public: \_\_sealed bool IsBypassed(Uri\* host);  
25      [VB] NotOverridable Public Function IsBypassed( ByVal host As Uri) As Boolean

1 [JScript] public function IsBypassed(host : Uri) : Boolean;

3 *Description*

4 Indicates whether to use the proxy server for the specified host.

5 *Return Value*: **true** if the proxy server should not be used for *host* ; otherwise,

6 **false** .

7 The **System.Net.WebProxy.IsBypassed(System.Uri)** method is used to  
8 determine whether to bypass the proxy server when accessing an Internet resource.

9 The **System.Uri** of the host to check for proxy use.

10 ISerializable.GetObjectData

11 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,

12 StreamingContext streamingContext);

13 [C++] void ISerializable::GetObjectData(SerializationInfo\* serializationInfo,

14 StreamingContext streamingContext);

15 [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal

16 streamingContext As StreamingContext) Implements ISerializable.GetObjectData

17 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,

18 streamingContext : StreamingContext);

20 WebRequest class (System.Net)

21 ToString

24 *Description*

1        Makes a request to a Uniform Resource Identifier (URI). This is an  
2        **abstract** class.

3        **System.Net.WebRequest** is the **abstract** base class for the .NET  
4        Framework's request/response model for accessing data from the Internet. An  
5        application that uses the request/response model can request data from the Internet  
6        in a protocol-agnostic manner, in which the application works with instances of  
7        the **System.Net.WebRequest** class while protocol-specific descendant classes  
8        carry out the details of the request.

9        **WebRequest**

10        *Example Syntax:*

11        **ToString**

13        [C#] protected WebRequest();

14        [C++] protected: WebRequest();

15        [VB] Protected Sub New()

16        [JScript] protected function WebRequest(); Initializes a new instance of the  
17        **System.Net.WebRequest** class.

19        *Description*

20        Initializes a new instance of the **System.Net.WebRequest** class.

21        Use the **System.Net.WebRequest.Create(System.Uri, System.Boolean)**  
22        method to initialize new **System.Net.WebRequest** instances. Do not use the  
23        constructor.

24        **WebRequest**

25        *Example Syntax:*

1       ToString

2

3       [C#] protected WebRequest(SerializationInfo serializationInfo, StreamingContext

4       streamingContext);

5       [C++] protected: WebRequest(SerializationInfo\* serializationInfo,

6       StreamingContext streamingContext);

7       [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal

8       streamingContext As StreamingContext)

9       [JScript] protected function WebRequest(serializationInfo : SerializationInfo,

10       streamingContext : StreamingContext);

11

12       *Description*

13       Initializes a new instance of the **System.Net.WebRequest** class from the

14       specified instances of the **System.Runtime.Serialization.SerializationInfo** and

15       **System.Runtime.Serialization.StreamingContext** classes.

16       When implemented by a descendant class, this constructor implements the

17       **System.Runtime.Serialization.ISerializable** interface for the

18       **System.Net.WebRequest** descendant. A

19       **System.Runtime.Serialization.SerializationInfo** that contains the information

20       required to serialize the new **System.Net.WebRequest** instance. A

21       **System.Runtime.Serialization.StreamingContext** that indicates the source of the

22       serialized stream associated with the new **System.Net.WebRequest** instance.

23       ConnectionGroupName

24       ToString

```
1
2 [C#] public virtual string ConnectionGroupName {get; set;}
3 [C++] public: __property virtual String* get_ConnectionGroupName();public:
4 __property virtual void set_ConnectionGroupName(String*);
5 [VB] Overridable Public Property ConnectionGroupName As String
6 [JScript] public function get ConnectionGroupName() : String;public function set
7 ConnectionGroupName(String);
```

#### *9 Description*

10 When overridden in a descendant class, gets or sets the name of the  
11 connection group for the request.

12 The **System.Net.WebRequest.ConnectionGroupName** property  
13 associates specific requests within an application to one or more connection pools.

14 ContentLength

15 ToString

```
16
17 [C#] public virtual long ContentLength {get; set;}
18 [C++] public: __property virtual __int64 get_ContentLength();public: __property
19 virtual void set_ContentLength(__int64);
20 [VB] Overridable Public Property ContentLength As Long
21 [JScript] public function get ContentLength() : long;public function set
22 ContentLength(long);
```

#### *24 Description*

1        When overridden in a descendant class, gets or sets the content length of  
2        the request data being sent.

3        The **System.Net.WebRequest.ContentLength** property contains the  
4        number of bytes of data sent to the Internet resource by the  
5        **System.Net.WebRequest** instance.

6        **ContentType**

7        **ToString**

8  
9        [C#] public virtual string ContentType {get; set;}

10        [C++] public: \_\_property virtual String\* get\_ContentType();public: \_\_property  
11        virtual void set\_ContentType(String\*);

12        [VB] Overridable Public Property ContentType As String

13        [JScript] public function get ContentType() : String;public function set  
14        ContentType(String);

15  
16        *Description*

17        When overridden in a descendant class, gets or sets the content type of the  
18        request data being sent.

19        The **System.Net.WebRequest.ContentType** property contains the media  
20        type of the request. This is typically the MIME encoding of the content.

21        **Credentials**

22        **ToString**

23  
24        [C#] public virtual ICredentials Credentials {get; set;}

25        [C++] public: \_\_property virtual ICredentials\* get\_Credentials();public:

```
1  __property virtual void set_Credentials(ICredentials*);  
2  [VB] Overridable Public Property Credentials As ICredentials  
3  [JScript] public function get Credentials() : ICredentials;public function set  
4  Credentials(ICredentials);  
5  
6  Description
```

7       When overridden in a descendant class, gets or sets the network credentials  
8       used for authenticating the request with the Internet resource.

9       The **System.Net.WebRequest.Credentials** property contains the  
10      authentication credentials required to access the Internet resource.

11      Headers

12      ToString

```
13  
14  [C#] public virtual WebHeaderCollection Headers {get; set;}  
15  [C++] public: __property virtual WebHeaderCollection* get_Headers();public:  
16  __property virtual void set_Headers(WebHeaderCollection*);  
17  [VB] Overridable Public Property Headers As WebHeaderCollection  
18  [JScript] public function get Headers() : WebHeaderCollection;public function set  
19  Headers(WebHeaderCollection);  
20  
21  Description
```

22       When overridden in a descendant class, gets or sets the collection of header  
23       name/value pairs associated with the request.

1        The **System.Net.WebRequest.Headers** property contains a  
2        **System.Net.WebHeaderCollection** instance containing the header information to  
3        send to the Internet resource.

4        Method

5        ToString

6

7        [C#] public virtual string Method {get; set;}  
8        [C++] public: \_\_property virtual String\* get\_Method();public: \_\_property virtual  
9        void set\_Method(String\*);  
10        [VB] Overridable Public Property Method As String  
11        [JScript] public function get Method() : String;public function set Method(String);

12

13        *Description*

14        When overridden in a descendant class, gets or sets the protocol method to  
15        use in this request.

16        When overridden in a descendant class, the  
17        **System.Net.WebRequest.Method** property contains the request method to use in  
18        this request.

19        PreAuthenticate

20        ToString

21

22        [C#] public virtual bool PreAuthenticate {get; set;}  
23        [C++] public: \_\_property virtual bool get\_PreAuthenticate();public: \_\_property  
24        virtual void set\_PreAuthenticate(bool);  
25        [VB] Overridable Public Property PreAuthenticate As Boolean

1 [JScript] public function get PreAuthenticate() : Boolean;public function set

2 PreAuthenticate(Boolean);

3

4 *Description*

5 When overridden in a descendant class, indicates whether to  
6 preauthenticate the request.

7 The **System.Net.WebRequest.PreAuthenticate** property indicates  
8 whether to send authentication information with the initial request. When  
9 **System.Net.WebRequest.PreAuthenticate** is **false** , the  
10 **System.Net.WebRequest** waits for an authentication challenge before sending  
11 authentication information.

12 **Proxy**

13 **ToString**

14

15 [C#] public virtual IWebProxy Proxy {get; set;}

16 [C++] public: \_\_property virtual IWebProxy\* get\_Proxy();public: \_\_property  
17 virtual void set\_Proxy(IWebProxy\*);

18 [VB] Overridable Public Property Proxy As IWebProxy

19 [JScript] public function get Proxy() : IWebProxy;public function set  
20 Proxy(IWebProxy);

21

22 *Description*

23 When overridden in a descendant class, gets or sets the network proxy to  
24 use to access this Internet resource.

1        The **System.Net.WebRequest.Proxy** property identifies the network proxy  
2        that the request uses to access the Internet resource. The request is made through  
3        the proxy server rather than directly to the Internet resource.

4        RequestUri

5        ToString

6  
7        [C#] public virtual Uri RequestUri {get;}  
8        [C++] public: \_\_property virtual Uri\* get\_RequestUri();  
9        [VB] Overridable Public ReadOnly Property RequestUri As Uri  
10        [JScript] public function get RequestUri() : Uri;

11  
12        *Description*

13        When overridden in a descendant class, gets the URI of the Internet  
14        resource associated with the request.

15        When overridden in a descendant class, the  
16        **System.Net.WebRequest.RequestUri** property contains the **System.Uri** instance  
17        that **System.Net.WebRequest.Create(System.Uri, System.Boolean)** method uses  
18        to create the request.

19        Timeout

20        ToString

21  
22        [C#] public virtual int Timeout {get; set;}  
23        [C++] public: \_\_property virtual int get\_Timeout(); public: \_\_property virtual void  
24        set\_Timeout(int);  
25        [VB] Overridable Public Property Timeout As Integer

1 [JScript] public function get Timeout() : int;public function set Timeout(int);

3 *Description*

4 Gets or sets the length of time before the request times out.

5 The **System.Net.WebRequest.Timeout** property indicates the length of  
6 time, in milliseconds, until the request times out and throws a  
7 **System.Net.WebException**. The **System.Net.WebRequest.Timeout** property  
8 affects only synchronous requests made with the  
9 **System.Net.WebRequest.GetResponse** method. To time out asynchronous  
10 requests, use the **System.Net.WebRequest.Abort** method.

11 **Abort**

13 [C#] public virtual void Abort();

14 [C++] public: virtual void Abort();

15 [VB] Overridable Public Sub Abort()

16 [JScript] public function Abort();

18 *Description*

19 Cancels an asynchronous request to an Internet resource.

20 The **System.Net.WebRequest.Abort** method cancels asynchronous  
21 requests to Internet resources started with the  
22 **System.Net.WebRequest.BeginGetResponse(System.AsyncCallback, System.O**  
23 **bject)** method.

24 **BeginGetRequestStream**

```
1  
2 [C#] public virtual IAsyncResult BeginGetRequestStream(AsyncCallback  
3 callback, object state);  
4 [C++] public: virtual IAsyncResult* BeginGetRequestStream(AsyncCallback*  
5 callback, Object* state);  
6 [VB] Overridable Public Function BeginGetRequestStream(ByVal callback As  
7 AsyncCallback, ByVal state As Object) As IAsyncResult  
8 [JScript] public function BeginGetRequestStream(callback : AsyncCallback, state  
9 : Object) : IAsyncResult;
```

### 11 *Description*

12 When overridden in a descendant class, provides an asynchronous version  
13 of the **System.Net.WebRequest.GetRequestStream** method.

14 *Return Value:* An **System.IAsyncResult** that references the asynchronous request.

15 The

16 **System.Net.WebRequest.BeginGetRequestStream(System.AsyncCallback, Sys**  
17 **tem.Object)** method starts an asynchronous request for a stream used to send data  
18 to an Internet resource. The callback method that implements the  
19 **System.AsyncCallback** delegate uses the  
20 **System.Net.WebRequest.EndGetRequestStream(System.IAsyncResult)**  
21 method to return the request stream. The **System.AsyncCallback** delegate. An  
22 object containing state information for this asynchronous request.

23 *BeginGetResponse*

```
24  
25 [C#] public virtual IAsyncResult BeginGetResponse(AsyncCallback callback,
```

```
1 object state);
2 [C++] public: virtual IAsyncResult* BeginGetResponse(AsyncCallback*
3 callback, Object* state);
4 [VB] Overridable Public Function BeginGetResponse( ByVal callback As
5 AsyncCallback, ByVal state As Object) As IAsyncResult
6 [JScript] public function BeginGetResponse(callback : AsyncCallback, state :
7 Object) : IAsyncResult;
```

8

9 *Description*

10       When overridden in a descendant class, begins an asynchronous request for
11 an Internet resource.

12 *Return Value:* An **System.IAsyncResult** that references the asynchronous request.

13       The

14 **System.Net.WebRequest.BeginGetResponse(System.AsyncCallback, System.O**
15 **bject)** method starts an asynchronous request for a response. The callback method
16 that implements the **System.AsyncCallback** delegate uses the
17 **System.Net.WebRequest.EndGetResponse(System.IAsyncResult)** method to
18 return the **System.Net.WebResponse** from the Internet resource. The
19 **System.AsyncCallback** delegate. An object containing state information for this
20 asynchronous request.

21       Create

22

23 [C#] public static WebRequest Create(string requestUriString);
24 [C++] public: static WebRequest\* Create(String\* requestUriString);
25 [VB] Public Shared Function Create( ByVal requestUriString As String) As

1      WebRequest

2      [JScript] public static function Create(requestUriString : String) : WebRequest;

3      Initializes a new **System.Net.WebRequest** .

4

5      *Description*

6          Initializes a new **System.Net.WebRequest** instance for the specified URI  
7          scheme.

8      *Return Value:* A **System.Net.WebRequest** descendant for the specific URI  
9          scheme.

10     The **System.Net.WebRequest.Create(System.Uri,System.Boolean)**  
11     method returns a descendant of the **System.Net.WebRequest** class determined at  
12     run time by the scheme of the URI in *requestUriString* . For example, when a URI  
13     beginning with http:// is passed in *requestUriString* , an  
14     **System.Net.HttpWebRequest** instance is returned by  
15     **System.Net.WebRequest.Create(System.Uri,System.Boolean)** . If a URI  
16     beginning with file:// is passed instead, the  
17     **System.Net.WebRequest.Create(System.Uri,System.Boolean)** method will  
18     return a **System.Net.FileWebRequest** instance. The URI that identifies the  
19     Internet resource.

20     Create

21

22     [C#] public static WebRequest Create(Uri requestUri);

23     [C++] public: static WebRequest\* Create(Uri\* requestUri);

24     [VB] Public Shared Function Create(ByVal requestUri As Uri) As WebRequest

25     [JScript] public static function Create(requestUri : Uri) : WebRequest;

1            *Description*

2                Initializes a new **System.Net.WebRequest** instance for the specified URI  
3                scheme.

4                *Return Value:* A **System.Net.WebRequest** descendant for the specified URI  
5                scheme.

6                The **System.Net.WebRequest.Create(System.Uri,Boolean)**  
7                method returns a descendant of the **System.Net.WebRequest** class determined at  
8                run time by the scheme of the URI in *requestUri* . For example, when a URI  
9                beginning with http:// is passed in *requestUri* , an **System.Net.HttpWebRequest**  
10                instance is returned by  
11                

12                **System.Net.WebRequest.Create(System.Uri,Boolean)** . If a URI  
13                beginning with file:// is passed instead, the  
14                **System.Net.WebRequest.Create(System.Uri,Boolean)** method will  
15                return a **System.Net.FileWebRequest** instance. A **System.Uri** containing the URI  
16                of the requested resource.

17                **CreateDefault**

18  
19                [C#] public static WebRequest CreateDefault(Uri requestUri);

20                [C++] public: static WebRequest\* CreateDefault(Uri\* requestUri);

21                [VB] Public Shared Function CreateDefault(ByVal requestUri As Uri) As

22                WebRequest

23                [JScript] public static function CreateDefault(requestUri : Uri) : WebRequest;

24  
25                *Description*

1       Initializes a new **System.Net.WebRequest** instance for the specified URI  
2       scheme.

3       *Return Value:* A **System.Net.WebRequest** descendant for the specified URI  
4       scheme.

5       The **System.Net.WebRequest.CreateDefault(System.Uri)** method returns  
6       a **System.Net.WebRequest** descendant instance based on only the scheme portion  
7       of a URI. A **System.Uri** containing the URI of the requested resource.

8       **EndGetRequestStream**

9  
10      [C#] public virtual Stream EndGetRequestStream(IAsyncResult asyncResult);  
11      [C++] public: virtual Stream\* EndGetRequestStream(IAsyncResult\*  
12        asyncResult);  
13      [VB] Overridable Public Function EndGetRequestStream(ByVal asyncResult As  
14        IAsyncResult) As Stream  
15      [JScript] public function EndGetRequestStream(asyncResult : IAsyncResult) :  
16        Stream;

17  
18       *Description*

19       When overridden in a descendant class, returns a **System.IO.Stream** for  
20       writing data to the Internet resource.

21       *Return Value:* A **System.IO.Stream** to write data to.

22       The  
23       **System.Net.WebRequest.EndGetRequestStream(System.IAsyncResult)**  
24       method completes an asynchronous stream request that was started by the  
25       **System.Net.WebRequest.BeginGetRequestStream(System.AsyncCallback,Sys**

1   **tem.Object)** method. An **System.IAsyncResult** that references a pending request  
2   for a stream.

3           **EndGetResponse**

4  
5   [C#] public virtual WebResponse EndGetResponse(IAsyncResult asyncResult);  
6   [C++] public: virtual WebResponse\* EndGetResponse(IAsyncResult\*  
7   asyncResult);  
8   [VB] Overridable Public Function EndGetResponse(ByVal asyncResult As  
9   IAsyncResult) As WebResponse  
10   [JScript] public function EndGetResponse(asyncResult : IAsyncResult) :  
11   WebResponse;

12  
13   *Description*

14       When overridden in a descendant class, returns a  
15   **System.Net.WebResponse** .

16   *Return Value:* A **System.Net.WebResponse** that contains a response to the  
17   Internet request.

18       The **System.Net.WebRequest.EndGetResponse(System.IAsyncResult)**  
19   method completes an asynchronous request for an Internet resource that was  
20   started with the  
21   **System.Net.WebRequest.BeginGetResponse(System.AsyncCallback, System.O**  
22   **bject)** method. An **System.IAsyncResult** that references a pending request for a  
23   response.

24           **GetRequestStream**

```
1  
2 [C#] public virtual Stream GetRequestStream();  
3 [C++] public: virtual Stream* GetRequestStream();  
4 [VB] Overridable Public Function GetRequestStream() As Stream  
5 [JScript] public function GetRequestStream() : Stream;  
6  
7 Description
```

When overridden in a descendant class, returns a **System.IO.Stream** for writing data to the Internet resource.

*Return Value:* A **System.IO.Stream** for writing data to the Internet resource.

The **System.Net.WebRequest.GetRequestStream** method initiates a request to send data to the Internet resource and returns a **System.IO.Stream** instance for sending data to the Internet resource.

#### GetResponse

```
14  
15  
16 [C#] public virtual WebResponse GetResponse();  
17 [C++] public: virtual WebResponse* GetResponse();  
18 [VB] Overridable Public Function GetResponse() As WebResponse  
19 [JScript] public function GetResponse() : WebResponse;  
20  
21 Description
```

When overridden in a descendant class, returns a response to an Internet request.

*Return Value:* A **System.Net.WebResponse** containing the response to the Internet request.

The **System.Net.WebRequest.GetResponse** method sends a request to an Internet resource and returns a **System.Net.WebResponse** instance. If the request has already been initiated by a call to **System.Net.WebRequest.GetRequestStream**, the **System.Net.WebRequest.GetResponse** method completes the request and returns any response.

## RegisterPrefix

```
[C#] public static bool RegisterPrefix(string prefix, IWebRequestCreate creator);  
[C++] public: static bool RegisterPrefix(String* prefix, IWebRequestCreate*  
creator);  
[VB] Public Shared Function RegisterPrefix(ByVal prefix As String, ByVal  
creator As IWebRequestCreate) As Boolean  
[JScript] public static function RegisterPrefix(prefix : String, creator :  
IWebRequestCreate) : Boolean;
```

### *Description*

Registers a **System.Net.WebRequest** descendant for the specified URI.

*Return Value:* **true** if registration is successful; otherwise, **false**.

The

**System.Net.WebRequest.RegisterPrefix(System.String, System.Net.IWebRequestCreate)** method registers **System.Net.WebRequest** descendants to service requests. The URI prefix that the **System.Net.WebRequest** descendant services. The create method that the **System.Net.WebRequest** calls to create the **System.Net.WebRequest** descendant.

```
1     ISerializable.GetObjectData  
2  
3 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,  
4 StreamingContext streamingContext);  
5  
6 [C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,  
7 StreamingContext streamingContext);  
8  
9 [VB] Sub GetObjectData( ByVal serializationInfo As SerializationInfo, ByVal  
10 streamingContext As StreamingContext) Implements ISerializable.GetObjectData  
11  
12 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,  
13 streamingContext : StreamingContext);  
14  
15     WebResponse class (System.Net)  
16  
17     ToString
```

#### *15 Description*

*16 Provides a response from a Uniform Resource Identifier (URI). This is an  
17 abstract class.*

*18 The **System.Net.WebResponse** class is the **abstract** base class from which  
19 protocol-specific response classes are derived. Applications can use the  
20 **System.Net.WebResponse** class to handle responses in a protocol-agnostic  
21 manner. In that case, the application works with instances of the  
22 **System.Net.WebResponse** class while the details of the request are carried out by  
23 protocol-specific descendant classes.*

*24 WebResponse*

*25 Example Syntax:*

1       ToString  
2  
3       [C#] protected WebResponse();  
4       [C++] protected: WebResponse();  
5       [VB] Protected Sub New()  
6       [JScript] protected function WebResponse(); Initializes a new instance of the  
7       **System.Net.WebResponse** class.  
8

9       *Description*

10       Initializes a new instance of the **System.Net.WebResponse** class.

11       Applications do not call the **System.Net.WebResponse** constructor  
12       directly, use the **System.Net.WebRequest.GetResponse** method on a  
13       **System.Net.WebRequest** instance.

14       WebResponse

15       *Example Syntax:*

16       ToString

17  
18       [C#] protected WebResponse(SerializationInfo serializationInfo,  
19       StreamingContext streamingContext);  
20       [C++] protected: WebResponse(SerializationInfo\* serializationInfo,  
21       StreamingContext streamingContext);  
22       [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal  
23       streamingContext As StreamingContext)  
24       [JScript] protected function WebResponse(serializationInfo : SerializationInfo,  
25       streamingContext : StreamingContext);

1  
2 *Description*

3       Initializes a new instance of the **System.Net.WebResponse** class from the  
4       specified instances of the **System.Runtime.Serialization.SerializationInfo** and  
5       **System.Runtime.Serialization.StreamingContext** classes.

6       When implemented by a descendant class, this constructor implements the  
7       **System.Runtime.Serialization.ISerializable** interface for the  
8       **System.Net.WebResponse** descendant. An instance of the  
9       **System.Runtime.Serialization.SerializationInfo** class containing the information  
10      required to serialize the new **System.Net.WebRequest** instance. An instance of  
11      the **System.Runtime.Serialization.StreamingContext** class indicating the source  
12      of the serialized stream associated with the new **System.Net.WebRequest**  
13      instance.

14       **ContentLength**

15       **ToString**

16  
17       [C#] public virtual long ContentLength {get; set;}

18       [C++] public: \_\_property virtual \_\_int64 get\_ContentLength();public: \_\_property  
19       virtual void set\_ContentLength(\_\_int64);

20       [VB] Overridable Public Property ContentLength As Long

21       [JScript] public function get ContentLength() : long;public function set  
22       ContentLength(long);

23  
24 *Description*

1        When overridden in a descendant class, gets or sets the content length of  
2 data being received.

3        The **System.Net.WebResponse.ContentLength** property contains the  
4 length, in bytes, of the response from the Internet resource. For request methods  
5 that contain header information, the **System.Net.WebResponse.ContentLength**  
6 does not include the length of the header information.

7        **ContentType**  
8        **ToString**

9  
10      [C#] public virtual string ContentType {get; set;}  
11      [C++] public: \_\_property virtual String\* get\_ContentType();public: \_\_property  
12      virtual void set\_ContentType(String\*);  
13      [VB] Overridable Public Property ContentType As String  
14      [JScript] public function get ContentType() : String;public function set  
15      ContentType(String);

16  
17      *Description*

18        When overridden in a derived class, gets or sets the content type of the data  
19 being received.

20        The **System.Net.WebResponse.ContentType** property contains the MIME  
21 content type of the response from the Internet resource, if known.

22        **Headers**  
23        **ToString**

24  
25      [C#] public virtual WebHeaderCollection Headers {get;}

```
1 [C++] public: __property virtual WebHeaderCollection* get_Headers();
2 [VB] Overridable Public ReadOnly Property Headers As WebHeaderCollection
3 [JScript] public function get Headers() : WebHeaderCollection;
```

#### 5 *Description*

6 When overridden in a derived class, gets a collection of header name-value  
7 pairs associated with this request.

8 The **System.Net.WebResponse.Headers** property contains the name-value  
9 header pairs returned in the response.

10 ResponseUri  
11 ToString

```
12
13 [C#] public virtual Uri ResponseUri {get;}
14 [C++] public: __property virtual Uri* get_ResponseUri();
15 [VB] Overridable Public ReadOnly Property ResponseUri As Uri
16 [JScript] public function get ResponseUri() : Uri;
```

#### 18 *Description*

19 When overridden in a derived class, gets the URI of the Internet resource  
20 that actually responded to the request.

21 The **System.Net.WebResponse.ResponseUri** property contains the URI of  
22 the Internet resource that actually provided the response data. This resource may  
23 not be the originally requested URI if the underlying protocol allows redirection of  
24 the request.

25 Close

```
1  
2 [C#] public virtual void Close();  
3 [C++] public: virtual void Close();  
4 [VB] Overridable Public Sub Close()  
5 [JScript] public function Close();  
6
```

7 *Description*

8 When overridden by a descendent class, closes the response stream.

9 The **System.Net.WebResponse.Close** method cleans up the resources used  
10 by a **System.Net.WebResponse** and closes the underlying stream by calling the  
11 **System.IO.Stream.Close** method.

12 **GetResponseStream**

```
13  
14 [C#] public virtual Stream GetResponseStream();  
15 [C++] public: virtual Stream* GetResponseStream();  
16 [VB] Overridable Public Function GetResponseStream() As Stream  
17 [JScript] public function GetResponseStream() : Stream;
```

18  
19 *Description*

20 When overridden in a descendant class, returns the data stream from the  
21 Internet resource.

22 *Return Value:* An instance of the **System.IO.Stream** class for reading data from  
23 the Internet resource.

24 The **System.Net.WebResponse.GetResponseStream** method returns the  
25 data stream from the Internet resource.

```
1  IDisposable.Dispose
2
3  [C#] void IDisposable.Dispose();
4  [C++] void IDisposable::Dispose();
5  [VB] Sub Dispose() Implements IDisposable.Dispose
6  [JScript] function IDisposable.Dispose();
7  ISerializable.GetObjectData
8
9  [C#] void ISeria
```

## System.Net.Sockets

The namespace provides a managed implementation of the Windows Sockets interface for developers that need to tightly control access to the network. Developers familiar with the Winsock API should have no problems developing applications using the class.

### *Description*

The **System.Net.Sockets** namespace provides a managed implementation of the Windows Sockets interface for developers that need to tightly control access to the network. Developers familiar with the Winsock API should have no problems developing applications using the **System.Net.Sockets.Socket** class.

AddressFamily enumeration (System.Net.Sockets)

### *Description*

1        Specifies the addressing scheme that an instance of the  
2        **System.Net.Sockets.Socket** class can use.

3        An **System.Net.Sockets.AddressFamily** member specifies the addressing  
4        scheme that a **System.Net.Sockets.Socket** instance uses to resolve an address. For  
5        example, **System.Net.Sockets.AddressFamily.InterNetwork** indicates that an IP  
6        version 4 address is expected when a **System.Net.Sockets.Socket** connects to an  
7        end point.

8  
9        [C#] public const AddressFamily AppleTalk;  
10      [C++] public: const AddressFamily AppleTalk;  
11      [VB] Public Const AppleTalk As AddressFamily  
12      [JScript] public var AppleTalk : AddressFamily;

13  
14      *Description*

15      AppleTalk address.

16  
17      [C#] public const AddressFamily Atm;  
18      [C++] public: const AddressFamily Atm;  
19      [VB] Public Const Atm As AddressFamily  
20      [JScript] public var Atm : AddressFamily;

21  
22      *Description*

23      Native ATM services address.

24  
25      [C#] public const AddressFamily Banyan;

1 [C++] public: const AddressFamily Banyan;  
2 [VB] Public Const Banyan As AddressFamily  
3 [JScript] public var Banyan : AddressFamily;

4

5 *Description*

6 Banyan address.

7

8 [C#] public const AddressFamily Ccitt;  
9 [C++] public: const AddressFamily Ccitt;  
10 [VB] Public Const Ccitt As AddressFamily  
11 [JScript] public var Ccitt : AddressFamily;

12

13 *Description*

14 Addresses for CCITT protocols, such as X.25.

15

16 [C#] public const AddressFamily Chaos;  
17 [C++] public: const AddressFamily Chaos;  
18 [VB] Public Const Chaos As AddressFamily  
19 [JScript] public var Chaos : AddressFamily;

20

21 *Description*

22 Address for MIT CHAOS protocols.

23

24 [C#] public const AddressFamily Cluster;  
25 [C++] public: const AddressFamily Cluster;

1 [VB] Public Const Cluster As AddressFamily

2 [JScript] public var Cluster : AddressFamily;

3  
4 *Description*

5 Address for Microsoft cluster products.

6  
7 [C#] public const AddressFamily DataKit;

8 [C++] public: const AddressFamily DataKit;

9 [VB] Public Const DataKit As AddressFamily

10 [JScript] public var DataKit : AddressFamily;

11  
12 *Description*

13 Address for Datakit protocols.

14  
15 [C#] public const AddressFamily DataLink;

16 [C++] public: const AddressFamily DataLink;

17 [VB] Public Const DataLink As AddressFamily

18 [JScript] public var DataLink : AddressFamily;

19  
20 *Description*

21 Direct data-link interface address.

22  
23 [C#] public const AddressFamily DecNet;

24 [C++] public: const AddressFamily DecNet;

25 [VB] Public Const DecNet As AddressFamily

1 [JScript] public var DecNet : AddressFamily;

3 *Description*

4 DECnet address.

6 [C#] public const AddressFamily Ecma;

7 [C++] public: const AddressFamily Ecma;

8 [VB] Public Const Ecma As AddressFamily

9 [JScript] public var Ecma : AddressFamily;

11 *Description*

12 European Computer Manufacturers Association (ECMA) address.

14 [C#] public const AddressFamily FireFox;

15 [C++] public: const AddressFamily FireFox;

16 [VB] Public Const FireFox As AddressFamily

17 [JScript] public var FireFox : AddressFamily;

19 *Description*

20 FireFox address.

22 [C#] public const AddressFamily HyperChannel;

23 [C++] public: const AddressFamily HyperChannel;

24 [VB] Public Const HyperChannel As AddressFamily

25 [JScript] public var HyperChannel : AddressFamily;

1           *Description*

2            NSC Hyperchannel address.

3            [C#] public const AddressFamily Ieee12844;

4            [C++] public: const AddressFamily Ieee12844;

5            [VB] Public Const Ieee12844 As AddressFamily

6            [JScript] public var Ieee12844 : AddressFamily;

7           *Description*

8            IEEE 1284.4 workgroup address.

9            [C#] public const AddressFamily ImpLink;

10           [C++] public: const AddressFamily ImpLink;

11           [VB] Public Const ImpLink As AddressFamily

12           [JScript] public var ImpLink : AddressFamily;

13           *Description*

14           ARPANET IMP address.

15           [C#] public const AddressFamily InterNetwork;

16           [C++] public: const AddressFamily InterNetwork;

17           [VB] Public Const InterNetwork As AddressFamily

18           [JScript] public var InterNetwork : AddressFamily;

1  
2 *Description*  
3     Address for IP version 4.  
4  
5 [C#] public const AddressFamily InterNetworkV6;  
6 [C++] public: const AddressFamily InterNetworkV6;  
7 [VB] Public Const InterNetworkV6 As AddressFamily  
8 [JScript] public var InterNetworkV6 : AddressFamily;

9  
10 *Description*  
11     Address for IP version 6.  
12  
13 [C#] public const AddressFamily Ipx;  
14 [C++] public: const AddressFamily Ipx;  
15 [VB] Public Const Ipx As AddressFamily  
16 [JScript] public var Ipx : AddressFamily;

17  
18 *Description*  
19     IPX or SPX address.  
20  
21 [C#] public const AddressFamily Irda;  
22 [C++] public: const AddressFamily Irda;  
23 [VB] Public Const Irda As AddressFamily  
24 [JScript] public var Irda : AddressFamily;  
25

1  
2 *Description*  
3       IrDA address.  
4  
5 [C#] public const AddressFamily Iso;  
6 [C++] public: const AddressFamily Iso;  
7 [VB] Public Const Iso As AddressFamily  
8 [JScript] public var Iso : AddressFamily;

9  
10 *Description*  
11       Address for ISO protocols.  
12  
13 [C#] public const AddressFamily Lat;  
14 [C++] public: const AddressFamily Lat;  
15 [VB] Public Const Lat As AddressFamily  
16 [JScript] public var Lat : AddressFamily;

17  
18 *Description*  
19       LAT address.  
20  
21 [C#] public const AddressFamily Max;  
22 [C++] public: const AddressFamily Max;  
23 [VB] Public Const Max As AddressFamily  
24 [JScript] public var Max : AddressFamily;

1           *Description*

2            MAX address.

5    [C#] public const AddressFamily NetBios;

6    [C++] public: const AddressFamily NetBios;

7    [VB] Public Const NetBios As AddressFamily

8    [JScript] public var NetBios : AddressFamily;

10          *Description*

11          NetBios address.

13    [C#] public const AddressFamily NetworkDesigners;

14    [C++] public: const AddressFamily NetworkDesigners;

15    [VB] Public Const NetworkDesigners As AddressFamily

16    [JScript] public var NetworkDesigners : AddressFamily;

18          *Description*

19          Address for Network Designers OSI gateway-enabled protocols.

21    [C#] public const AddressFamily NS;

22    [C++] public: const AddressFamily NS;

23    [VB] Public Const NS As AddressFamily

24    [JScript] public var NS : AddressFamily;

1  
2     *Description*  
3         Address for Xerox NS protocols.

4  
5     [C#] public const AddressFamily Osi;  
6     [C++] public: const AddressFamily Osi;  
7     [VB] Public Const Osi As AddressFamily  
8     [JScript] public var Osi : AddressFamily;

9  
10    *Description*  
11         Address for ISO protocols.

12  
13     [C#] public const AddressFamily Pup;  
14     [C++] public: const AddressFamily Pup;  
15     [VB] Public Const Pup As AddressFamily  
16     [JScript] public var Pup : AddressFamily;

17  
18    *Description*  
19         Address for PUP protocols.

20  
21     [C#] public const AddressFamily Sna;  
22     [C++] public: const AddressFamily Sna;  
23     [VB] Public Const Sna As AddressFamily  
24     [JScript] public var Sna : AddressFamily;

1           *Description*

2            IBM SNA address.

5    [C#] public const AddressFamily Unix;

6    [C++] public: const AddressFamily Unix;

7    [VB] Public Const Unix As AddressFamily

8    [JScript] public var Unix : AddressFamily;

10          *Description*

11            Unix local to host address.

13    [C#] public const AddressFamily Unknown;

14    [C++] public: const AddressFamily Unknown;

15    [VB] Public Const Unknown As AddressFamily

16    [JScript] public var Unknown : AddressFamily;

18          *Description*

19            Unknown address family.

21    [C#] public const AddressFamily Unspecified;

22    [C++] public: const AddressFamily Unspecified;

23    [VB] Public Const Unspecified As AddressFamily

24    [JScript] public var Unspecified : AddressFamily;

1  
2 *Description*  
3     Unspecified address family.  
4  
5 [C#] public const AddressFamily VoiceView;  
6 [C++] public: const AddressFamily VoiceView;  
7 [VB] Public Const VoiceView As AddressFamily  
8 [JScript] public var VoiceView : AddressFamily;

9  
10 *Description*  
11     VoiceView address.  
12     Methods:  
13       LingerOption class (System.Net.Sockets)  
14       ToString

15  
16  
17 *Description*  
18     Contains information about a socket's linger time, the amount of time it will  
19     remain after closing if data remains to be sent.  
20     The settings of the **System.Net.Sockets.LingerOption** instance associated  
21     with a **System.Net.Sockets.Socket** instance control the length of time that a  
22     socket will remain after closing if data remains to be sent.

23     Constructors:  
24       LingerOption

25     *Example Syntax:*

```
1  ToString
2
3  [C#] public LingerOption(bool enable, int seconds);
4  [C++] public: LingerOption(bool enable, int seconds);
5  [VB] Public Sub New(ByVal enable As Boolean, ByVal seconds As Integer)
6  [JScript] public function LingerOption(enable : Boolean, seconds : int);
```

```
7
8 Description
```

```
9     Initializes a new instance of the System.Net.Sockets.LingerOption class.
10
11    The System.Net.Sockets.LingerOption instance is created with
12    System.Net.Sockets.LingerOption.Enabled property set to enable and the
13    System.Net.Sockets.LingerOption.LingerTime property set to seconds . true to
14    enable remaining connected after System.Net.Sockets.Socket.Close is called;
15    otherwise, false . The number of seconds to remain connected after
16    System.Net.Sockets.Socket.Close is called.
```

```
17
18     Properties:
19
20     Enabled
21
22     ToString
23
24
```

```
25
26 [C#] public bool Enabled {get; set;}
27 [C++] public: __property bool get_Enabled();public: __property void
28 set_Enabled(bool);
29 [VB] Public Property Enabled As Boolean
30 [JScript] public function get Enabled() : Boolean;public function set
31 Enabled(Boolean);
```

1           *Description*

2           Gets or sets a value indicating whether to linger after the socket is closed.

3           LingerTime

4           ToString

5

6

7 [C#] public int LingerTime {get; set;}

8 [C++] public: \_\_property int get\_LingerTime();public: \_\_property void  
9 set\_LingerTime(int);

10 [VB] Public Property LingerTime As Integer

11 [JScript] public function get LingerTime() : int;public function set  
12 LingerTime(int);

13

14           *Description*

15           Gets or sets the amount of time to remain connected after the socket is  
16 closed.

17           MulticastOption class (System.Net.Sockets)

18           ToString

19

20

21           *Description*

22           Contains IP address values for IP multicast packets.

23           The **System.Net.Sockets.MulticastOption** class sets IP address values  
24 when joining or leaving an IP multicast group. It is used with the  
25 **System.Net.Sockets.Socket.SetSocketOption(System.Net.Sockets.SocketOptio**

1    **nLevel, System.Net.Sockets.SocketOptionName, System.Int32)** and  
2    **System.Net.Sockets.Socket.GetSocketOption(System.Net.Sockets.SocketOptio**  
3    **nLevel, System.Net.Sockets.SocketOptionName)** methods when the *optionName*  
4    parameter is set to **System.Net.Sockets.SocketOptionName.AddMembership** or  
5    **System.Net.Sockets.SocketOptionName.DropMembership** .

6    MulticastOption

7    *Example Syntax:*

8    ToString

9  
10    [C#] public MulticastOption(IPAddress group);  
11    [C++] public: MulticastOption(IPAddress\* group);  
12    [VB] Public Sub New( ByVal group As IPAddress)  
13    [JScript] public function MulticastOption(group : IPAddress);

14  
15    *Description*

16       Initializes a new version of the **System.Net.Sockets.MulticastOption** class  
17    for the specified IP multicast group. The IP address of the multicast group.

18    MulticastOption

19    *Example Syntax:*

20    ToString

21  
22    [C#] public MulticastOption(IPAddress group, IPAddress mcint);  
23    [C++] public: MulticastOption(IPAddress\* group, IPAddress\* mcint);  
24    [VB] Public Sub New( ByVal group As IPAddress, ByVal mcint As IPAddress)  
25    [JScript] public function MulticastOption(group : IPAddress, mcint : IPAddress);

1     Initializes a new instance of the **System.Net.Sockets.MulticastOption** class.

3     *Description*

4         Initializes a new instance of the **System.Net.Sockets.MulticastOption**  
5         class with the specified IP multicast group address and local interface address. The  
6         group IP address. The local IP address.

7             Group

8             ToString

10        [C#] public IPAddress Group {get; set;}

11        [C++] public: \_\_property IPAddress\* get\_Group(); public: \_\_property void  
12           set\_Group(IPAddress\*);

13        [VB] Public Property Group As IPAddress

14        [JScript] public function get Group() : IPAddress; public function set  
15           Group(IPAddress);

17     *Description*

18         Gets or sets the IP address of a multicast group.

19         Valid IP addresses for multicast packets are in the range 224.0.0.0 to  
20           239.255.255.255.

21             LocalAddress

22             ToString

24        [C#] public IPAddress LocalAddress {get; set;}

25        [C++] public: \_\_property IPAddress\* get\_LocalAddress(); public: \_\_property void

```
1  set_LocalAddress(IPAddress*);  
2  [VB] Public Property LocalAddress As IPAddress  
3  [JScript] public function get LocalAddress() : IPAddress;public function set  
4  LocalAddress(IPAddress);  
5  
6  Description
```

Gets or sets the local address associated with a multicast group.

The **System.Net.Sockets.MulticastOption.LocalAddress** property contains the IP address of the interface associated with the multicast group membership. If **System.Net.Sockets.MulticastOption.LocalAddress** is set to **System.Net.IPAddress.Any** , the default interface is used.

NetworkStream class (System.Net.Sockets)

ToString

*Description*

Provides the underlying stream of data for network access.

**System.Net.Sockets.NetworkStream** implements the standard .NET Framework stream mechanism to send and receive data through network sockets.

NetworkStream

*Example Syntax:*

ToString

```
24 [C#] public NetworkStream(Socket socket);  
25 [C++] public: NetworkStream(Socket* socket);
```

1 [VB] Public Sub New(ByVal socket As Socket)  
2 [JScript] public function NetworkStream(socket : Socket); Creates a new instance  
3 of the **System.Net.Sockets.NetworkStream** class.

5 *Description*

6 Creates a new instance of the **System.Net.Sockets.NetworkStream** class  
7 for the specified **System.Net.Sockets.Socket** .

8 The **System.Net.Sockets.NetworkStream** is created with read/write access  
9 to the specified socket. The **System.Net.Sockets.NetworkStream** does not own  
10 the underlying socket, so calling the **System.Net.Sockets.NetworkStream.Close**  
11 method will not close the socket. The **System.Net.Sockets.Socket** that provides  
12 the network data for the new **System.Net.Sockets.NetworkStream** .

13 NetworkStream

14 *Example Syntax:*

15 ToString

17 [C#] public NetworkStream(Socket socket, bool ownsSocket);

18 [C++] public: NetworkStream(Socket\* socket, bool ownsSocket);

19 [VB] Public Sub New(ByVal socket As Socket, ByVal ownsSocket As Boolean)

20 [JScript] public function NetworkStream(socket : Socket, ownsSocket : Boolean);

22 *Description*

23 Initializes a new instance of the **System.Net.Sockets.NetworkStream** class  
24 for the specified socket with the specified **System.Net.Sockets.Socket** ownership.

1        The **System.Net.Sockets.NetworkStream** is created with read/write access  
2 to the specified socket. If *ownsSocket* is **true** , the  
3 **System.Net.Sockets.NetworkStream** owns the underlying socket, and calling  
4 **System.Net.Sockets.NetworkStream.Close** will also close the underlying socket.  
5 The network socket that the new **System.Net.Sockets.NetworkStream** will  
6 encapsulate. **true** if the socket will be owned by this  
7 **System.Net.Sockets.NetworkStream** instance; otherwise, **false**.

8        **NetworkStream**

9        *Example Syntax:*

10      **ToString**

11  
12     [C#] public NetworkStream(Socket socket, FileAccess access);  
13     [C++] public: NetworkStream(Socket\* socket, FileAccess access);  
14     [VB] Public Sub New(ByVal socket As Socket, ByVal access As FileAccess)  
15     [JScript] public function NetworkStream(socket : Socket, access : FileAccess);

16  
17      *Description*

18        Creates a new instance of the **System.Net.Sockets.NetworkStream** class  
19 for the specified **System.Net.Sockets.Socket** with the specified access rights.

20        The **System.Net.Sockets.NetworkStream** is created with the specified  
21 access to the specified socket. The **System.Net.Sockets.NetworkStream** does not  
22 own the underlying socket, so calling the  
23 **System.Net.Sockets.NetworkStream.Close** method will not close the socket. The  
24 **System.Net.Sockets.Socket** that provides the network data. One of the  
25 **System.IO.FileAccess** values that sets the

1      **System.Net.Sockets.NetworkStream.CanRead** and  
2      **System.Net.Sockets.NetworkStream.CanWrite** properties of the  
3      **System.Net.Sockets.NetworkStream** .  
4      **NetworkStream**  
5      *Example Syntax:*  
6      **ToString**  
7  
8      [C#] public NetworkStream(Socket socket, FileAccess access, bool ownsSocket);  
9      [C++] public: NetworkStream(Socket\* socket, FileAccess access, bool  
10     ownsSocket);  
11     [VB] Public Sub New(ByVal socket As Socket, ByVal access As FileAccess,  
12     ByVal ownsSocket As Boolean)  
13     [JScript] public function NetworkStream(socket : Socket, access : FileAccess,  
14     ownsSocket : Boolean);  
15  
16     *Description*  
17       Creates a new instance of the **System.Net.Sockets.NetworkStream** class  
18       for the specified **System.Net.Sockets.Socket** with the specified access rights and  
19       the specified **System.Net.Sockets.Socket** ownership.  
20       The **System.Net.Sockets.NetworkStream** is created with the specified  
21       access to the specified socket. If *ownsSocket* is **true** , the  
22       **System.Net.Sockets.NetworkStream** owns the underlying socket, and calling  
23       **System.Net.Sockets.NetworkStream.Close** will also close the underlying socket.  
24       The **System.Net.Sockets.Socket** that provides the network data. One of the  
25       **System.IO.FileAccess** values that sets the

1   **System.Net.Sockets.NetworkStream.CanRead** and  
2   **System.Net.Sockets.NetworkStream.CanWrite** properties of the  
3   **System.Net.Sockets.NetworkStream** . **true** if the socket will be owned by this  
4   **System.Net.Sockets.NetworkStream** instance; otherwise, **false**.

5           CanRead

6           ToString

7  
8   [C#] public override bool CanRead {get;}

9   [C++] public: \_\_property virtual bool get\_CanRead();

10   [VB] Overrides Public ReadOnly Property CanRead As Boolean

11   [JScript] public function get CanRead() : Boolean;

12  
13   *Description*

14       Gets a value indicating whether the current stream supports reading.

15           CanSeek

16           ToString

17  
18   [C#] public override bool CanSeek {get;}

19   [C++] public: \_\_property virtual bool get\_CanSeek();

20   [VB] Overrides Public ReadOnly Property CanSeek As Boolean

21   [JScript] public function get CanSeek() : Boolean;

22  
23   *Description*

24       Gets a value indicating whether the stream supports seeking. This property

25       always returns **false** .

1           CanWrite  
2           ToString  
3  
4 [C#] public override bool CanWrite {get;}  
5 [C++] public: \_\_property virtual bool get\_CanWrite();  
6 [VB] Overrides Public ReadOnly Property CanWrite As Boolean  
7 [JScript] public function get CanWrite() : Boolean;  
8

9 *Description*

10         Gets a value that indicates whether the current stream supports writing.

11         DataAvailable

12         ToString

13  
14 [C#] public virtual bool DataAvailable {get;}  
15 [C++] public: \_\_property virtual bool get\_DataAvailable();  
16 [VB] Overridable Public ReadOnly Property DataAvailable As Boolean  
17 [JScript] public function get DataAvailable() : Boolean;

18  
19 *Description*

20         Gets a value indicating whether data is available on the stream to be read.

21         Length

22         ToString

23  
24 [C#] public override long Length {get;}  
25 [C++] public: \_\_property virtual \_\_int64 get\_Length();

1 [VB] Overrides Public ReadOnly Property Length As Long  
2 [JScript] public function get Length() : long;

3  
4 *Description*

5 The length of the data available on the stream. This property always throws  
6 a **System.NotSupportedException** .

7 Position  
8 ToString

9  
10 [C#] public override long Position {get; set;}  
11 [C++] public: \_\_property virtual \_\_int64 get\_Position();public: \_\_property virtual  
12 void set\_Position(\_\_int64);

13 [VB] Overrides Public Property Position As Long

14 [JScript] public function get Position() : long;public function set Position(long);

15  
16 *Description*

17 Gets or sets the current position in the stream. This property always throws  
18 a **System.NotSupportedException** .

19 Readable  
20 ToString

21  
22 [C#] protected bool Readable {get; set;}

23 [C++] protected: \_\_property bool get\_Readable();protected: \_\_property void  
24 set\_Readable(bool);

25 [VB] Protected Property Readable As Boolean

1 [JScript] protected function get Readable() : Boolean;protected function set  
2 Readable(Boolean);

3

4 *Description*

5 Gets or sets a value indicating that the stream can be read.

6 The **System.Net.Sockets.NetworkStream.Readable** property is accessible  
7 only through this class or a derived class. Other classes can determine whether a  
8 **System.Net.Sockets.NetworkStream** is readable by checking the  
9 **System.Net.Sockets.NetworkStream.CanRead** property.

10     Socket

11     ToString

12

13 [C#] protected Socket Socket {get;}

14 [C++] protected: \_\_property Socket\* get\_Socket();

15 [VB] Protected ReadOnly Property Socket As Socket

16 [JScript] protected function get Socket() : Socket;

17

18 *Description*

19 Gets the underlying network connection.

20 The **System.Net.Sockets.NetworkStream.Socket** property represents the  
21 underlying network socket that this **System.Net.Sockets.NetworkStream**  
22 encapsulates.

23     Writeable

24     ToString

25

```
1
2 [C#] protected bool Writeable {get; set;}
3 [C++] protected: __property bool get_Writeable();protected: __property void
4 set_Writeable(bool);
5 [VB] Protected Property Writeable As Boolean
6 [JScript] protected function get Writeable() : Boolean;protected function set
7 Writeable(Boolean);
8
```

#### 9 *Description*

10 Gets a value that indicates whether the stream is writable.

11 The **System.Net.Sockets.NetworkStream.Writeable** property is  
12 accessible only through this class or a derived class. Other classes can determine  
13 whether a **System.Net.Sockets.NetworkStream** is writable by checking the  
14 **System.Net.Sockets.NetworkStream.CanWrite** property.

#### 15 BeginRead

```
16
17 [C#] public override IAsyncResult BeginRead(byte[] buffer, int offset, int size,
18 AsyncCallback callback, object state);
19 [C++] public: IAsyncResult* BeginRead(unsigned char buffer __gc[], int offset,
20 int size, AsyncCallback* callback, Object* state);
21 [VB] Overrides Public Function BeginRead(ByVal buffer() As Byte, ByVal offset
22 As Integer, ByVal size As Integer, ByVal callback As AsyncCallback, ByVal state
23 As Object) As IAsyncResult
24 [JScript] public override function BeginRead(buffer : Byte[], offset : int, size : int,
25 callback : AsyncCallback, state : Object) : IAsyncResult;
```

1  
2 *Description*

3       Begins an asynchronous read from a stream.

4 *Return Value:* An **System.IAsyncResult** representing the asynchronous call.

5       The

6 **System.Net.Sockets.NetworkStream.BeginRead(System.Byte[],System.Int32,S**

7 **ystem.Int32,System.AsyncCallback,System.Object)** method begins an

8 asynchronous request for data from a network stream. The method that

9 implements the **System.AsyncCallback** delegate uses the

10 **System.Net.Sockets.NetworkStream.EndRead(System.IAsyncResult)** method

11 to return the amount of data read from the stream. The location in memory that

12 stores the data from the stream. The location in *buffer* to begin storing the data to.

13 The size of *buffer*. The delegate to call when the asynchronous call is complete.

14 An object containing additional information supplied by the client.

15       BeginWrite

16  
17 [C#] public override IAsyncResult BeginWrite(byte[] buffer, int offset, int size,  
18 AsyncCallback callback, object state);

19 [C++] public: IAsyncResult\* BeginWrite(unsigned char buffer \_\_gc[], int offset,  
20 int size, AsyncCallback\* callback, Object\* state);

21 [VB] Overrides Public Function BeginWrite( ByVal buffer() As Byte, ByVal offset  
22 As Integer, ByVal size As Integer, ByVal callback As AsyncCallback, ByVal state  
23 As Object) As IAsyncResult

24 [JScript] public override function BeginWrite(buffer : Byte[], offset : int, size : int,  
25 callback : AsyncCallback, state : Object) : IAsyncResult;

1            *Description*

2            Begins an asynchronous write to a stream.

3            *Return Value:* An **System.IAsyncResult** representing the asynchronous call.

4            The

5            **System.Net.Sockets.NetworkStream.BeginWrite(System.Byte[],System.Int32,**

6            **System.Int32,System.AsyncCallback,System.Object)** method begins to send

7            data asynchronously to the network stream. The method that implements the

8            **System.AsyncCallback** delegate uses the

9            **System.Net.Sockets.NetworkStream.EndWrite(System.IAsyncResult)** method

10           to complete sending the data. The location in memory that holds the data to send.

11           The location in *buffer* to begin sending the data. The size of *buffer* . The delegate

12           to call when the asynchronous call is complete. An object containing additional

13           information supplied by the client.

14           Close

15           [C#] public override void Close();

16           [C++] public: void Close();

17           [VB] Overrides Public Sub Close()

18           [JScript] public override function Close();

19           *Description*

20           Closes the stream and optionally closes the underlying socket.

21           The **System.Net.Sockets.NetworkStream.Close** method frees resources

22           used by the **System.Net.Sockets.NetworkStream** instance and, if the

1   **System.Net.Sockets.NetworkStream** owns the underlying socket, closes the  
2   underlying socket.

3    **Dispose**

4  
5   [C#] protected virtual void Dispose(bool disposing);  
6   [C++] protected; virtual void Dispose(bool disposing);  
7   [VB] Overridable Protected Sub Dispose(ByVal disposing As Boolean)  
8   [JScript] protected function Dispose(disposing : Boolean);

9  
10   *Description*

11   Cleans up a network stream. **true** if this method was called by another  
12   method such as **System.Net.Sockets.NetworkStream.Close** or  
13   **System.Net.Sockets.NetworkStream.Dispose(System.Boolean)**; **false** if this  
14   method was called by the finalizer.

15   **EndRead**

16  
17   [C#] public override int EndRead(IAsyncResult asyncResult);  
18   [C++] public: int EndRead(IAsyncResult\* asyncResult);  
19   [VB] Overrides Public Function EndRead(ByVal asyncResult As IAsyncResult)  
20   As Integer  
21   [JScript] public override function EndRead(asyncResult : IAsyncResult) : int;

22  
23   *Description*

24   Handles the end of an asynchronous read.

25   *Return Value:* The number of bytes read from the stream.

1       The  
2 **System.Net.Sockets.NetworkStream.EndRead(System.IAsyncResult)** method  
3 completes an asynchronous read of the network stream started with the  
4 **System.Net.Sockets.NetworkStream.BeginRead(System.Byte[],System.Int32,S**  
5 **ystem.Int32,System.AsyncCallback,System.Object)** method. An  
6 **System.IAsyncResult** representing an asynchronous call.

7       **EndWrite**

8  
9 [C#] public override void EndWrite(IAsyncResult asyncResult);  
10 [C++] public: void EndWrite(IAsyncResult\* asyncResult);  
11 [VB] Overrides Public Sub EndWrite(ByVal asyncResult As IAsyncResult)  
12 [JScript] public override function EndWrite(asyncResult : IAsyncResult);  
13

14 *Description*

15       Handles the end of an asynchronous write.

16       The

17 **System.Net.Sockets.NetworkStream.EndWrite(System.IAsyncResult)** method  
18 completes an asynchronous write to a network stream started with the  
19 **System.Net.Sockets.NetworkStream.BeginWrite(System.Byte[],System.Int32,**  
20 **System.Int32,System.AsyncCallback,System.Object)** method. The  
21 **System.IAsyncResult** representing the asynchronous call.

22       **Finalize**

23  
24 [C#] ~NetworkStream();  
25 [C++] ~NetworkStream();

1 [VB] Overrides Protected Sub Finalize()  
2 [JScript] protected override function Finalize();

3  
4 *Description*

5       Frees resources used by the **System.Net.Sockets.NetworkStream** .

6       The **System.Net.Sockets.NetworkStream** finalizer calls the  
7 **System.Net.Sockets.NetworkStream.Close** method to close the stream.

8       Flush

9  
10 [C#] public override void Flush();

11 [C++] public: void Flush();

12 [VB] Overrides Public Sub Flush()

13 [JScript] public override function Flush();

14  
15 *Description*

16       Flushes data from the stream. This method is reserved for future use.

17       The **System.Net.Sockets.NetworkStream.Flush** method implements the  
18 **System.IO.Stream.Flush** method but, because  
19 **System.Net.Sockets.NetworkStream** is not buffered, has no effect on network  
20 streams. Calling the **System.Net.Sockets.NetworkStream.Flush** method will not  
21 throw an exception.

22       Read

23  
24 [C#] public override int Read(in byte[] buffer, int offset, int size);

25 [C++] public: int Read(\_\_in unsigned char\* buffer \_\_gc[], int offset, int size);

1 [VB] Overrides Public Function Read( ByVal buffer() As Byte, ByVal offset As  
2 Integer, ByVal size As Integer) As Integer  
3 [JScript] public override function Read( in buffer : Byte[], offset : int, size : int) :  
4 int;

5  
6 *Description*

7     Reads data from the stream.

8     *Return Value*: The number of bytes read from the stream.

9     The

10 **System.Net.Sockets.NetworkStream.Read(System.Byte[],System.Int32, System**  
11 **.Int32)** method reads data from the network stream into a data buffer. The location  
12 in memory to store data read from the stream. The location in the buffer to begin  
13 storing the data to. The number of bytes to read from the stream.

14     *Seek*

15  
16 [C#] public override long Seek(long offset, SeekOrigin origin);

17 [C++] public: \_\_int64 Seek(\_\_int64 offset, SeekOrigin origin);

18 [VB] Overrides Public Function Seek( ByVal offset As Long, ByVal origin As  
19 SeekOrigin) As Long

20 [JScript] public override function Seek(offset : long, origin : SeekOrigin) : long;

21  
22 *Description*

23     Sets the current position of the stream to the given value. This method  
24 always throws a **System.NotSupportedException**. This parameter is not used.  
25 This parameter is not used.

```
1  SetLength
2
3  [C#] public override void SetLength(long value);
4  [C++] public: void SetLength(__int64 value);
5  [VB] Overrides Public Sub SetLength(ByVal value As Long)
6  [JScript] public override function SetLength(value : long);
```

```
7
8 Description
```

```
9     Sets the length of the stream. This method always throws a
10    System.NotSupportedException . This parameter is not used.
```

```
11   IDisposable.Dispose
```

```
12
13  [C#] void IDisposable.Dispose();
14  [C++] void IDisposable::Dispose();
15  [VB] Sub Dispose() Implements IDisposable.Dispose
16  [JScript] function IDisposable.Dispose();
```

```
17   Write
```

```
18
19  [C#] public override void Write(byte[] buffer, int offset, int size);
20  [C++] public: void Write(unsigned char buffer __gc[], int offset, int size);
21  [VB] Overrides Public Sub Write(ByVal buffer() As Byte, ByVal offset As
22    Integer, ByVal size As Integer)
23  [JScript] public override function Write(buffer : Byte[], offset : int, size : int);
```

```
24
25 Description
```

Writes data to the stream.

*Return Value:* The number of bytes written to the stream.

The

## System.Net.Sockets.NetworkStream.Write(System.Byte[], System.Int32, System.Int32)

`m.Int32`) method sends the contents of a data buffer to the network. The data to

write to the stream. The location in the buffer to start writing data from. The

number of bytes to write to the stream.

## ProtocolFamily enumeration (System.Net.Sockets)

## WriteByte

### *Description*

Specifies the type of protocol that an instance of the

**System.Net.Sockets.Socket** class can use.

The **System.Net.Sockets.ProtocolFamily** enumeration specifies the protocol scheme used by the **System.Net.Sockets.Socket** class to resolve an address. For example, **System.Net.Sockets.ProtocolFamily.InterNetwork** indicates that the IP version 4 protocol is expected when a

**System.Net.Sockets.Socket** connects to an endpoint.

## WriteByte

```
[C#] public const ProtocolFamily AppleTalk;
```

[C++] public: const ProtocolFamily AppleTalk;

[VB] Public Const AppleTalk As ProtocolFamily

[JScript] public var AppleTalk : ProtocolFamily;

1           *Description*

2           AppleTalk protocol.

3           WriteByte

4

5

6 [C#] public const ProtocolFamily Atm;

7 [C++] public: const ProtocolFamily Atm;

8 [VB] Public Const Atm As ProtocolFamily

9 [JScript] public var Atm : ProtocolFamily;

10

11 *Description*

12           Native ATM services protocol.

13           WriteByte

14

15 [C#] public const ProtocolFamily Banyan;

16 [C++] public: const ProtocolFamily Banyan;

17 [VB] Public Const Banyan As ProtocolFamily

18 [JScript] public var Banyan : ProtocolFamily;

19

20 *Description*

21           Banyan protocol.

22           WriteByte

23

24 [C#] public const ProtocolFamily Ccitt;

25 [C++] public: const ProtocolFamily Ccitt;

1 [VB] Public Const Ccitt As ProtocolFamily  
2 [JScript] public var Ccitt : ProtocolFamily;

3  
4 *Description*

5 CCITT protocol, such as X.25.

6 WriteByte

7  
8 [C#] public const ProtocolFamily Chaos;  
9 [C++] public: const ProtocolFamily Chaos;  
10 [VB] Public Const Chaos As ProtocolFamily  
11 [JScript] public var Chaos : ProtocolFamily;

12  
13 *Description*

14 MIT CHAOS protocol.

15 WriteByte

16  
17 [C#] public const ProtocolFamily Cluster;  
18 [C++] public: const ProtocolFamily Cluster;  
19 [VB] Public Const Cluster As ProtocolFamily  
20 [JScript] public var Cluster : ProtocolFamily;

21  
22 *Description*

23 Microsoft Cluster products protocol.

24 WriteByte

25

```
1
2 [C#] public const ProtocolFamily DataKit;
3 [C++] public: const ProtocolFamily DataKit;
4 [VB] Public Const DataKit As ProtocolFamily
5 [JScript] public var DataKit : ProtocolFamily;
```

7 *Description*

8 DataKit protocol.

9 WriteByte

```
10
11 [C#] public const ProtocolFamily DataLink;
12 [C++] public: const ProtocolFamily DataLink;
13 [VB] Public Const DataLink As ProtocolFamily
14 [JScript] public var DataLink : ProtocolFamily;
```

16 *Description*

17 Direct data link protocol.

18 WriteByte

```
19
20 [C#] public const ProtocolFamily DecNet;
21 [C++] public: const ProtocolFamily DecNet;
22 [VB] Public Const DecNet As ProtocolFamily
23 [JScript] public var DecNet : ProtocolFamily;
```

25 *Description*

1 DECNet protocol.  
2 WriteByte  
3  
4 [C#] public const ProtocolFamily Ecma;  
5 [C++] public: const ProtocolFamily Ecma;  
6 [VB] Public Const Ecma As ProtocolFamily  
7 [JScript] public var Ecma : ProtocolFamily;  
8

9 *Description*

10 European Computer Manufacturers Association (ECMA) protocol.  
11 WriteByte  
12

13 [C#] public const ProtocolFamily FireFox;  
14 [C++] public: const ProtocolFamily FireFox;  
15 [VB] Public Const FireFox As ProtocolFamily  
16 [JScript] public var FireFox : ProtocolFamily;  
17

18 *Description*

19 FireFox protocol.  
20 WriteByte  
21

22 [C#] public const ProtocolFamily HyperChannel;  
23 [C++] public: const ProtocolFamily HyperChannel;  
24 [VB] Public Const HyperChannel As ProtocolFamily  
25 [JScript] public var HyperChannel : ProtocolFamily;

1           *Description*

2            NSC HyperChannel protocol.

3            WriteByte

4

5

6 [C#] public const ProtocolFamily Ieee12844;

7 [C++] public: const ProtocolFamily Ieee12844;

8 [VB] Public Const Ieee12844 As ProtocolFamily

9 [JScript] public var Ieee12844 : ProtocolFamily;

10

11           *Description*

12           IEEE 1284.4 workgroup protocol.

13           WriteByte

14

15 [C#] public const ProtocolFamily ImpLink;

16 [C++] public: const ProtocolFamily ImpLink;

17 [VB] Public Const ImpLink As ProtocolFamily

18 [JScript] public var ImpLink : ProtocolFamily;

19

20           *Description*

21           ARPANET IMP protocol.

22           WriteByte

23

24 [C#] public const ProtocolFamily InterNetwork;

25 [C++] public: const ProtocolFamily InterNetwork;

1 [VB] Public Const InterNetwork As ProtocolFamily  
2 [JScript] public var InterNetwork : ProtocolFamily;

3  
4 *Description*

5 IP version 4 protocol.

6 WriteByte

7  
8 [C#] public const ProtocolFamily InterNetworkV6;

9 [C++] public: const ProtocolFamily InterNetworkV6;

10 [VB] Public Const InterNetworkV6 As ProtocolFamily

11 [JScript] public var InterNetworkV6 : ProtocolFamily;

12  
13 *Description*

14 IP version 6 protocol.

15 WriteByte

16  
17 [C#] public const ProtocolFamily Ipx;

18 [C++] public: const ProtocolFamily Ipx;

19 [VB] Public Const Ipx As ProtocolFamily

20 [JScript] public var Ipx : ProtocolFamily;

21  
22 *Description*

23 IPX or SPX protocol.

24 WriteByte

1  
2 [C#] public const ProtocolFamily Irda;  
3 [C++] public: const ProtocolFamily Irda;  
4 [VB] Public Const Irda As ProtocolFamily  
5 [JScript] public var Irda : ProtocolFamily;

6  
7 *Description*

8 IrDA protocol.

9 WriteByte

10  
11 [C#] public const ProtocolFamily Iso;  
12 [C++] public: const ProtocolFamily Iso;  
13 [VB] Public Const Iso As ProtocolFamily  
14 [JScript] public var Iso : ProtocolFamily;

15  
16 *Description*

17 ISO protocol.

18 WriteByte

19  
20 [C#] public const ProtocolFamily Lat;  
21 [C++] public: const ProtocolFamily Lat;  
22 [VB] Public Const Lat As ProtocolFamily  
23 [JScript] public var Lat : ProtocolFamily;

24  
25 *Description*

1 LAT protocol.  
2 WriteByte  
3  
4 [C#] public const ProtocolFamily Max;  
5 [C++] public: const ProtocolFamily Max;  
6 [VB] Public Const Max As ProtocolFamily  
7 [JScript] public var Max : ProtocolFamily;  
8

9 *Description*

10 MAX protocol.  
11 WriteByte  
12  
13 [C#] public const ProtocolFamily NetBios;  
14 [C++] public: const ProtocolFamily NetBios;  
15 [VB] Public Const NetBios As ProtocolFamily  
16 [JScript] public var NetBios : ProtocolFamily;  
17

18 *Description*

19 NetBios protocol.  
20 WriteByte  
21  
22 [C#] public const ProtocolFamily NetworkDesigners;  
23 [C++] public: const ProtocolFamily NetworkDesigners;  
24 [VB] Public Const NetworkDesigners As ProtocolFamily  
25 [JScript] public var NetworkDesigners : ProtocolFamily;

## 2 | Description

Network Designers OSI gateway enabled protocol.

## WriteByte

[C#] public const ProtocolFamily NS;

[C++] public: const ProtocolFamily NS;

[VB] Public Const NS As ProtocolFamily

[JScript] public var NS : ProtocolFamily;

### *Description*

Xerox NS protocol.

## WriteByte

```
[C#] public const ProtocolFamily Osi;
```

[C++] public: const ProtocolFamily Osi;

[VB] Public Const Osi As ProtocolFamily

[JScript] public var Osi : ProtocolFamily;

### *Description*

OSI protocol.

## WriteByte

[C#] public const ProtocolFamily Pup;

[C++] public: const ProtocolFamily Pup;

1 [VB] Public Const Pup As ProtocolFamily  
2 [JScript] public var Pup : ProtocolFamily;

3  
4 *Description*

5 PUP protocol.

6 WriteByte

7  
8 [C#] public const ProtocolFamily Sna;

9 [C++] public: const ProtocolFamily Sna;

10 [VB] Public Const Sna As ProtocolFamily

11 [JScript] public var Sna : ProtocolFamily;

12  
13 *Description*

14 IBM SNA protocol.

15 WriteByte

16  
17 [C#] public const ProtocolFamily Unix;

18 [C++] public: const ProtocolFamily Unix;

19 [VB] Public Const Unix As ProtocolFamily

20 [JScript] public var Unix : ProtocolFamily;

21  
22 *Description*

23 Unix local to host protocol.

24 WriteByte

25

1  
2 [C#] public const ProtocolFamily Unknown;  
3 [C++] public: const ProtocolFamily Unknown;  
4 [VB] Public Const Unknown As ProtocolFamily  
5 [JScript] public var Unknown : ProtocolFamily;

6  
7 *Description*

8 Unknown protocol.

9 WriteByte

10  
11 [C#] public const ProtocolFamily Unspecified;  
12 [C++] public: const ProtocolFamily Unspecified;  
13 [VB] Public Const Unspecified As ProtocolFamily  
14 [JScript] public var Unspecified : ProtocolFamily;

15  
16 *Description*

17 Unspecified protocol.

18 WriteByte

19  
20 [C#] public const ProtocolFamily VoiceView;  
21 [C++] public: const ProtocolFamily VoiceView;  
22 [VB] Public Const VoiceView As ProtocolFamily  
23 [JScript] public var VoiceView : ProtocolFamily;

24  
25 *Description*

1      VoiceView protocol.

2      **ProtocolType** enumeration (System.Net.Sockets)

3      **ToString**

4

5

6      *Description*

7      Specifies the protocols that the **System.Net.Sockets.Socket** class supports.

8      The **System.Net.Sockets.ProtocolType** enumeration is used by the

9

10

11     **System.Net.Sockets.Socket** class to indicate to the Windows Socket API the

12     requested protocol for the socket. Low-level driver software for the requested

13     protocol must be present on the computer for the **System.Net.Sockets.Socket** to

14

15

16     **ToString**

17

18

19     [C#] public const ProtocolType Ggp;

20     [C++] public: const ProtocolType Ggp;

21     [VB] Public Const Ggp As ProtocolType

22     [JScript] public var Ggp : ProtocolType;

23

24

25     *Description*

26     Gateway To Gateway Protocol.

27     **ToString**

28

29

30     [C#] public const ProtocolType Icmp;

31     [C++] public: const ProtocolType Icmp;

32

33

1 [VB] Public Const Icmp As ProtocolType  
2 [JScript] public var Icmp : ProtocolType;

3  
4 *Description*

5 Internet Control Message Protocol.

6 ToString

7  
8 [C#] public const ProtocolType Idp;  
9 [C++] public: const ProtocolType Idp;  
10 [VB] Public Const Idp As ProtocolType  
11 [JScript] public var Idp : ProtocolType;

12  
13 *Description*

14 IDP Protocol.

15 ToString

16  
17 [C#] public const ProtocolType Igmp;  
18 [C++] public: const ProtocolType Igmp;  
19 [VB] Public Const Igmp As ProtocolType  
20 [JScript] public var Igmp : ProtocolType;

21  
22 *Description*

23 Internet Group Management Protocol.

24 ToString

25

1  
2 [C#] public const ProtocolType IP;  
3 [C++] public: const ProtocolType IP;  
4 [VB] Public Const IP As ProtocolType  
5 [JScript] public var IP : ProtocolType;

6  
7 *Description*

8 Internet Protocol.

9 **ToString**

10  
11 [C#] public const ProtocolType Ipx;  
12 [C++] public: const ProtocolType Ipx;  
13 [VB] Public Const Ipx As ProtocolType  
14 [JScript] public var Ipx : ProtocolType;

15  
16 *Description*

17 IPX Protocol.

18 **ToString**

19  
20 [C#] public const ProtocolType ND;  
21 [C++] public: const ProtocolType ND;  
22 [VB] Public Const ND As ProtocolType  
23 [JScript] public var ND : ProtocolType;

24  
25 *Description*

1                   Net Disk Protocol (unofficial).  
2                   ToString  
3  
4 [C#] public const ProtocolType Pup;  
5 [C++] public: const ProtocolType Pup;  
6 [VB] Public Const Pup As ProtocolType  
7 [JScript] public var Pup : ProtocolType;  
8

9                   *Description*

10                  PUP Protocol.

11                  ToString  
12  
13 [C#] public const ProtocolType Raw;  
14 [C++] public: const ProtocolType Raw;  
15 [VB] Public Const Raw As ProtocolType  
16 [JScript] public var Raw : ProtocolType;  
17

18                   *Description*

19                  Raw UP packet protocol.

20                  ToString  
21  
22 [C#] public const ProtocolType Spx;  
23 [C++] public: const ProtocolType Spx;  
24 [VB] Public Const Spx As ProtocolType  
25 [JScript] public var Spx : ProtocolType;

1      *Description*

2            SPX Protocol.

3      *ToString*

4

5

6 [C#] public const ProtocolType SpxII;

7 [C++] public: const ProtocolType SpxII;

8 [VB] Public Const SpxII As ProtocolType

9 [JScript] public var SpxII : ProtocolType;

10

11 *Description*

12            SPX Version 2 Protocol.

13      *ToString*

14

15 [C#] public const ProtocolType Tcp;

16 [C++] public: const ProtocolType Tcp;

17 [VB] Public Const Tcp As ProtocolType

18 [JScript] public var Tcp : ProtocolType;

19

20 *Description*

21            Transmission Control Protocol.

22      *ToString*

23

24 [C#] public const ProtocolType Udp;

25 [C++] public: const ProtocolType Udp;

1    [VB] Public Const Udp As ProtocolType  
2    [JScript] public var Udp : ProtocolType;  
3  
4    *Description*  
5       User Datagram Protocol.  
6    ToString  
7  
8    [C#] public const ProtocolType Unknown;  
9    [C++] public: const ProtocolType Unknown;  
10   [VB] Public Const Unknown As ProtocolType  
11   [JScript] public var Unknown : ProtocolType;  
12  
13   *Description*  
14       Unknown protocol.  
15   ToString  
16  
17   [C#] public const ProtocolType Unspecified;  
18   [C++] public: const ProtocolType Unspecified;  
19   [VB] Public Const Unspecified As ProtocolType  
20   [JScript] public var Unspecified : ProtocolType;  
21  
22   *Description*  
23       Unspecified protocol.  
24       SelectMode enumeration (System.Net.Sockets)  
25   ToString

1  
2  
3 *Description*

4       Defines the polling modes for the

5       **System.Net.Sockets.Socket.Poll(System.Int32, System.Net.Sockets.SelectMode**  
6       **)** method.

7       The **System.Net.Sockets.SelectMode** enumeration defines the polling  
8       modes that can be passed to the

9       **System.Net.Sockets.Socket.Poll(System.Int32, System.Net.Sockets.SelectMode**  
10      **)** method.

11       **ToString**

12  
13      [C#] public const SelectMode SelectError;

14      [C++] public: const SelectMode SelectError;

15      [VB] Public Const SelectError As SelectMode

16      [JScript] public var SelectError : SelectMode;

17  
18 *Description*

19       Error status mode.

20       **ToString**

21  
22      [C#] public const SelectMode SelectRead;

23      [C++] public: const SelectMode SelectRead;

24      [VB] Public Const SelectRead As SelectMode

25      [JScript] public var SelectRead : SelectMode;

1           *Description*

2            Read status mode.

3           *ToString*

4

5

6 [C#] public const SelectMode SelectWrite;

7 [C++] public: const SelectMode SelectWrite;

8 [VB] Public Const SelectWrite As SelectMode

9 [JScript] public var SelectWrite : SelectMode;

10

11           *Description*

12           Write status mode.

13           Socket class (System.Net.Sockets)

14           *ToString*

15

16

17           *Description*

18           Implements the Berkeley sockets interface.

19           The **System.Net.Sockets.Socket** class creates a managed version of an

20 Internet transport service. Once the socket is created, it is bound to a specific

21 endpoint through the **System.Net.Sockets.Socket.Bind(System.NetEndPoint)**

22 method, and the connection to that endpoint is established through the

23 **System.Net.Sockets.Socket.Connect(System.NetEndPoint)** method.

24           **Socket**

25           *Example Syntax:*

1       ToString

2

3       [C#] public Socket(AddressFamily addressFamily, SocketType socketType,  
4       ProtocolType protocolType);

5       [C++] public: Socket(AddressFamily addressFamily, SocketType socketType,  
6       ProtocolType protocolType);

7       [VB] Public Sub New(ByVal addressFamily As AddressFamily, ByVal  
8       socketType As SocketType, ByVal protocolType As ProtocolType)  
9       [JScript] public function Socket(addressFamily : AddressFamily, socketType :  
10       SocketType, protocolType : ProtocolType);

11

12       *Description*

13       Initializes a new instance of the **System.Net.Sockets.Socket** class.  
14       The *addressFamily* parameter specifies the addressing scheme that the  
15       socket uses, the *socketType* parameter specifies the type of the socket, and the  
16       *protocolType* parameter specifies the protocol used by the socket. The three  
17       parameters are not independent. Some address families restrict which protocols  
18       can be used with them, and often the socket type is implicit in the protocol. If the  
19       combination of address family, socket type, and protocol type results in an invalid  
20       socket, a **System.Net.Sockets.SocketException** is thrown. One of the  
21       **System.Net.Sockets.AddressFamily** values. One of the  
22       **System.Net.Sockets.SocketType** values. One of the  
23       **System.Net.Sockets.ProtocolType** values.

24       AddressFamily

25       ToString

```
1
2 [C#] public AddressFamily AddressFamily {get;}
3 [C++] public: __property AddressFamily get_AddressFamily();
4 [VB] Public ReadOnly Property AddressFamily As AddressFamily
5 [JScript] public function get AddressFamily() : AddressFamily;
```

#### *7 Description*

8 Gets the address family of the socket.

9 Available

10 ToString

```
11
12 [C#] public int Available {get;}
13 [C++] public: __property int get_Available();
14 [VB] Public ReadOnly Property Available As Integer
15 [JScript] public function get Available() : int;
```

#### *17 Description*

18 Gets the amount of data that has been received from the network and is  
19 available to be read.

20 Blocking

21 ToString

```
22
23 [C#] public bool Blocking {get; set;}
24 [C++] public: __property bool get_Blocking(); public: __property void
25 set_Blocking(bool);
```

1 [VB] Public Property Blocking As Boolean  
2 [JScript] public function get Blocking() : Boolean;public function set  
3 Blocking(Boolean);

5 *Description*

6 Gets or sets a value that indicates whether the socket is in blocking mode.

7 The **System.Net.Sockets.Socket.Blocking** property indicates whether a  
8 **System.Net.Sockets.Socket** is in blocking mode.

9 Connected

10 ToString

12 [C#] public bool Connected {get;}

13 [C++] public: \_\_property bool get\_Connected();

14 [VB] Public ReadOnly Property Connected As Boolean

15 [JScript] public function get Connected() : Boolean;

17 *Description*

18 Gets a value indicating whether a socket is connected to a remote resource.

19 The **System.Net.Sockets.Socket.Connected** returns **true** if the socket has  
20 either connected to a remote resource or accepted a connection from a remote  
21 resource.

22 Handle

23 ToString

25 [C#] public IntPtr Handle {get;}

```
1 [C++] public: __property IntPtr get_Handle();  
2 [VB] Public ReadOnly Property Handle As IntPtr  
3 [JScript] public function get Handle() : IntPtr;  
4  
5
```

#### *5 Description*

6 Gets the operating system handle for the socket.

7 LocalEndPoint

8 ToString

```
9  
10 [C#] public EndPoint LocalEndPoint {get;}  
11 [C++] public: __property EndPoint* get_LocalEndPoint();  
12 [VB] Public ReadOnly Property LocalEndPoint As EndPoint  
13 [JScript] public function get LocalEndPoint() : EndPoint;  
14  
15
```

#### *15 Description*

16 Gets the local endpoint.

17 The **System.Net.Sockets.Socket.LocalEndPoint** property contains the  
18 network connection information associated with the local network device.

19 ProtocolType

20 ToString

```
21  
22 [C#] public ProtocolType ProtocolType {get;}  
23 [C++] public: __property ProtocolType get_ProtocolType();  
24 [VB] Public ReadOnly Property ProtocolType As ProtocolType  
25 [JScript] public function get ProtocolType() : ProtocolType;
```

1           *Description*

2           Gets the protocol type of the socket.

3           **RemoteEndPoint**

4           **ToString**

5

6

7 [C#] public EndPoint RemoteEndPoint {get;}

8 [C++] public: \_\_property EndPoint\* get\_RemoteEndPoint();

9 [VB] Public ReadOnly Property RemoteEndPoint As EndPoint

10 [JScript] public function get RemoteEndPoint() : EndPoint;

11           *Description*

12           Gets the remote endpoint.

13           The **System.Net.Sockets.Socket.RemoteEndPoint** property gets the  
14           network connection information associated with the remote network device.

15           **SocketType**

16           **ToString**

17

18

19 [C#] public SocketType SocketType {get;}

20 [C++] public: \_\_property SocketType get\_SocketType();

21 [VB] Public ReadOnly Property SocketType As SocketType

22 [JScript] public function get SocketType() : SocketType;

23

24           *Description*

25           Gets the type of the socket.

```
1     Accept  
2  
3 [C#] public Socket Accept();  
4 [C++] public: Socket* Accept();  
5 [VB] Public Function Accept() As Socket  
6 [JScript] public function Accept() : Socket;  
7  
8 Description
```

Creates a new **System.Net.Sockets.Socket** to handle an incoming connection request.

*Return Value:* A **System.Net.Sockets.Socket** instance to handle an incoming connection request.

The **System.Net.Sockets.Socket.Accept** method extracts the first connection request from the queue of pending requests and creates a new **System.Net.Sockets.Socket** instance to handle it. The **System.Net.Sockets.Socket** returned by **System.Net.Sockets.Socket.Accept** is a duplicate of the current **System.Net.Sockets.Socket**.

BeginAccept

```
19  
20 [C#] public IAsyncResult BeginAccept(AsyncCallback callback, object state);  
21 [C++] public: IAsyncResult* BeginAccept(AsyncCallback* callback, Object*  
22 state);  
23 [VB] Public Function BeginAccept(ByVal callback As AsyncCallback, ByVal  
24 state As Object) As IAsyncResult  
25 [JScript] public function BeginAccept(callback : AsyncCallback, state : Object) :
```

1 IAsyncResult;

3 *Description*

4 Begins an asynchronous request to create a new socket to accept an  
5 incoming connection request.

6 *Return Value:* An **System.IAsyncResult** that references the asynchronous socket  
7 creation.

8 The

9 **System.Net.Sockets.Socket.BeginAccept(System.AsyncCallback, System.Object)**  
10 method starts an asynchronous request to create a socket to handle an incoming  
11 connection request. The callback method that implements the  
12 **System.AsyncCallback** delegate uses the  
13 **System.Net.Sockets.Socket.EndAccept(System.IAsyncResult)** method to return  
14 the socket. The **System.AsyncCallback** delegate is an object containing state  
15 information for this request.

16 BeginConnect

17  
18 [C#] public IAsyncResult BeginConnect(EndPoint remoteEP, AsyncCallback  
19 callback, object state);

20 [C++] public: IAsyncResult\* BeginConnect(EndPoint\* remoteEP,  
21 AsyncCallback\* callback, Object\* state);

22 [VB] Public Function BeginConnect(ByVal remoteEP As EndPoint, ByVal  
23 callback As AsyncCallback, ByVal state As Object) As IAsyncResult

24 [JScript] public function BeginConnect(remoteEP : EndPoint, callback :  
25 AsyncCallback, state : Object) : IAsyncResult;

1  
2 *Description*

3       Begins an asynchronous request for a connection to a network device.

4 *Return Value:* An **System.IAsyncResult** that references the asynchronous  
5 connection.

6       The  
7 **System.Net.Sockets.Socket.BeginConnect(System.NetEndPoint, System.Async**  
8 **Callback, System.Object)** method starts an asynchronous request for a connection  
9 to a remote device. The callback method that implements the  
10 **System.AsyncCallback** delegate uses the  
11 **System.Net.Sockets.Socket.EndConnect(System.IAsyncResult)** method to  
12 return the socket. An **System.NetEndPoint** instance that represents the remote  
13 device. The **System.AsyncCallback** delegate. An object that contains state  
14 information for this request.

15       BeginReceive

16  
17 [C#] public IAsyncResult BeginReceive(byte[] buffer, int offset, int size,

18       SocketFlags socketFlags, AsyncCallback callback, object state);

19 [C++] public: IAsyncResult\* BeginReceive(unsigned char buffer \_\_gc[], int  
20       offset, int size, SocketFlags socketFlags, AsyncCallback\* callback, Object\* state);

21 [VB] Public Function BeginReceive(ByVal buffer() As Byte, ByVal offset As  
22       Integer, ByVal size As Integer, ByVal socketFlags As SocketFlags, ByVal  
23       callback As AsyncCallback, ByVal state As Object) As IAsyncResult

24 [JScript] public function BeginReceive(buffer : Byte[], offset : int, size : int,  
25       socketFlags : SocketFlags, callback : AsyncCallback, state : Object) :

1 IAsyncResult;

3 *Description*

4 Begins to asynchronously receive data from a connected socket.

5 *Return Value:* An **System.IAsyncResult** that references the asynchronous read.

6 The

7 **System.Net.Sockets.Socket.BeginReceive(System.Byte[],System.Int32, System.**  
8 **Int32, System.Net.Sockets.SocketFlags, System.AsyncCallback, System.Object)**

9 method starts asynchronously reading data from a socket. The callback method

10 that implements the **System.AsyncCallback** delegate uses the

11 **System.Net.Sockets.Socket.EndReceive(System.IAsyncResult)** method to  
12 return the data read from the socket. The storage location for the received data.

13 The zero-based position in the *buffer* at which to store the received data. The  
14 number of bytes to receive. A bitwise combination of the

15 **System.Net.Sockets.SocketFlags** values. The **System.AsyncCallback** delegate.

16 An object containing state information for this request.

17 **BeginReceiveFrom**

19 [C#] public IAsyncResult BeginReceiveFrom(byte[] buffer, int offset, int size,

20 SocketFlags socketFlags, refEndPoint remoteEP, AsyncCallback callback, object  
21 state);

22 [C++] public: IAsyncResult\* BeginReceiveFrom(unsigned char buffer \_\_gc[], int  
23 offset, int size, SocketFlags socketFlags, EndPoint\*\* remoteEP, AsyncCallback\*  
24 callback, Object\* state);

25 [VB] Public Function BeginReceiveFrom(ByVal buffer() As Byte, ByVal offset

```
1 As Integer, ByVal size As Integer, ByVal socketFlags As SocketFlags, ByRef
2 remoteEP As EndPoint, ByVal callback As AsyncCallback, ByVal state As
3 Object) As IAsyncResult
4 [JScript] public function BeginReceiveFrom(buffer : Byte[], offset : int, size : int,
5 socketFlags : SocketFlags, remoteEP : EndPoint, callback : AsyncCallback, state :
6 Object) : IAsyncResult;
```

7

8 *Description*

9 Begins to asynchronously receive data from a specified network device.

10 *Return Value:* An **System.IAsyncResult** that references the asynchronous read.

11 The

12 **System.Net.Sockets.Socket.BeginReceiveFrom(System.Byte[],System.Int32,Sy**  
13 **stem.Int32,System.Net.Sockets.SocketFlags,System.Net.EndPoint@,System.A**  
14 **syncCallback,System.Object)** method starts asynchronously reading data from a  
15 socket. The callback method that implements the **System.AsyncCallback** delegate  
16 uses the **System.Net.Sockets.Socket.EndReceive(System.IAsyncResult)** method  
17 to return the data read from the socket. The storage location for the received data.  
18 The zero-based position in the *buffer* at which to store the data. The number of  
19 bytes to receive. A bitwise combination of the **System.Net.Sockets.SocketFlags**  
20 values. An **System.Net.EndPoint** that represents the source of the data. The  
21 **System.AsyncCallback** delegate. An object containing state information for this  
22 request.

23 *BeginSend*

24

25 [C#] public IAsyncResult BeginSend(byte[] buffer, int offset, int size, SocketFlags

```
1  socketFlags, AsyncCallback callback, object state);  
2  [C++] public: IAsyncResult* BeginSend(unsigned char buffer __gc[], int offset,  
3  int size, SocketFlags socketFlags, AsyncCallback* callback, Object* state);  
4  [VB] Public Function BeginSend(ByVal buffer() As Byte, ByVal offset As  
5  Integer, ByVal size As Integer, ByVal socketFlags As SocketFlags, ByVal  
6  callback As AsyncCallback, ByVal state As Object) As IAsyncResult  
7  [JScript] public function BeginSend(buffer : Byte[], offset : int, size : int,  
8  socketFlags : SocketFlags, callback : AsyncCallback, state : Object) :  
9  IAsyncResult;
```

10

11 *Description*

12     Sends data asynchronously to a connected socket.

13 *Return Value:* An **System.IAsyncResult** that references the asynchronous send.

14     The

15 **System.Net.Sockets.Socket.BeginSend(System.Byte[],System.Int32,System.Int**  
16 **32,System.Net.Sockets.SocketFlags,System.AsyncCallback,System.Object)**

17     method starts asynchronously sending data through a socket. The callback method

18     that implements the **System.AsyncCallback** delegate uses the

19 **System.Net.Sockets.Socket.EndSend(System.IAsyncResult)** method to

20     complete sending data. The data to send. The zero-based position in *buffer* at

21     which to begin sending data. The number of bytes to send. A bitwise combination

22     of the **System.Net.Sockets.SocketFlags** values. The **System.AsyncCallback**

23     delegate. An object containing state information for this request.

24     BeginSendTo

25

```
1  
2 [C#] public IAsyncResult BeginSendTo(byte[] buffer, int offset, int size,  
3 SocketFlags socketFlags, EndPoint remoteEP, AsyncCallback callback, object  
4 state);  
5 [C++] public: IAsyncResult* BeginSendTo(unsigned char buffer __gc[], int  
6 offset, int size, SocketFlags socketFlags, EndPoint* remoteEP, AsyncCallback*  
7 callback, Object* state);  
8 [VB] Public Function BeginSendTo(ByVal buffer() As Byte, ByVal offset As  
9 Integer, ByVal size As Integer, ByVal socketFlags As SocketFlags, ByVal  
10 remoteEP As EndPoint, ByVal callback As AsyncCallback, ByVal state As  
11 Object) As IAsyncResult  
12 [JScript] public function BeginSendTo(buffer : Byte[], offset : int, size : int,  
13 socketFlags : SocketFlags, remoteEP : EndPoint, callback : AsyncCallback, state :  
14 Object) : IAsyncResult;  
15
```

#### 16 *Description*

17 Sends data asynchronously to a specific network device.

18 *Return Value:* An **System.IAsyncResult** that references the asynchronous send.

19 The

20 **System.Net.Sockets.Socket.BeginSendTo(System.Byte[],System.Int32,System.**  
21 **Int32, System.Net.Sockets.SocketFlags, System.Net.EndPoint, System.AsyncCal**  
22 **lback, System.Object)** method starts asynchronously sending data through a  
23 socket. The callback method that implements the **System.AsyncCallback** delegate  
24 uses the **System.Net.Sockets.Socket.EndSendTo(System.IAsyncResult)** method  
25 to complete sending data. The data to send. The zero-based position in *buffer* at

1 which to begin sending data. The number of bytes to send. A bitwise combination  
2 of the **System.Net.Sockets.SocketFlags** values. An **System.NetEndPoint** that  
3 represents the remote device. The **System.AsyncCallback** delegate. An object  
4 containing state information for this request.

5 **Bind**

6

7 [C#] public void Bind(EndPoint localEP);  
8 [C++] public: void Bind(EndPoint\* localEP);  
9 [VB] Public Sub Bind( ByVal localEP AsEndPoint)  
10 [JScript] public function Bind(localEP :EndPoint);

11

12 *Description*

13     Associates a socket with a local endpoint.

14     You must call the

15     **System.Net.Sockets.Socket.Bind(System.NetEndPoint)** method before you call  
16     the **System.Net.Sockets.Socket.Listen(System.Int32)** or  
17     **System.Net.Sockets.Socket.Connect(System.NetEndPoint)** methods. The local  
18     **System.NetEndPoint** to associate with the socket.

19     **Close**

20

21 [C#] public void Close();  
22 [C++] public: void Close();  
23 [VB] Public Sub Close()  
24 [JScript] public function Close();

1  
2 *Description*

3       Forces a socket connection to close.

4       The **System.Net.Sockets.Socket.Connected** property is set to **false** when  
5       the socket is closed.

6       **Connect**

7  
8       [C#] public void Connect(EndPoint remoteEP);

9       [C++] public: void Connect(EndPoint\* remoteEP);

10      [VB] Public Sub Connect( ByVal remoteEP As EndPoint)

11      [JScript] public function Connect(remoteEP : EndPoint);

12  
13 *Description*

14       Establishes a connection to a remote device.

15       The **System.Net.Sockets.Socket.Connect(System.NetEndPoint)** method  
16       establishes a network connection between

17       **System.Net.Sockets.Socket.LocalEndPoint** and the device identified by  
18       *remoteEP* . Once the connection has been made, you can send data to the remote  
19       device with the  
20       **System.Net.Sockets.Socket.Send(System.Byte[],System.Int32,System.Net.Sock**  
21       **ets.SocketFlags)** method, or receive data from the remote device with the  
22       **System.Net.Sockets.Socket.Receive(System.Byte[],System.Int32,System.Net.S**  
23       **ockets.SocketFlags)** method. An **System.NetEndPoint** instance that represents  
24       the remote device.

25       **Dispose**

1  
2 [C#] protected virtual void Dispose(bool disposing);  
3 [C++] protected: virtual void Dispose(bool disposing);  
4 [VB] Overridable Protected Sub Dispose(ByVal disposing As Boolean)  
5 [JScript] protected function Dispose(disposing : Boolean);  
6

7 *Description*

8  
9 EndAccept

10  
11 [C#] public Socket EndAccept(IAsyncResult asyncResult);  
12 [C++] public: Socket\* EndAccept(IAsyncResult\* asyncResult);  
13 [VB] Public Function EndAccept( ByVal asyncResult As IAsyncResult ) As Socket  
14 [JScript] public function EndAccept(asyncResult : IAsyncResult ) : Socket;

15  
16 *Description*

17       Ends an asynchronous request to create a new socket to accept an incoming  
18 connection request.

19       *Return Value:* A **System.Net.Sockets.Socket** to handle the incoming connection.

20       The **System.Net.Sockets.Socket.EndAccept(System.IAsyncResult)**  
21 method completes a request for a connection that was started with the  
22 **System.Net.Sockets.Socket.BeginAccept(System.AsyncCallback, System.Object)**  
23 method. The pending request for an incoming socket.

24  
25 EndConnect

```
1  
2 [C#] public void EndConnect(IAsyncResult asyncResult);  
3 [C++] public: void EndConnect(IAsyncResult* asyncResult);  
4 [VB] Public Sub EndConnect(ByVal asyncResult As IAsyncResult)  
5 [JScript] public function EndConnect(asyncResult : IAsyncResult);  
6  
7 Description
```

8       Ends a pending synchronous connection request. The pending  
9       asynchronous connection request.

10      EndReceive

```
11  
12 [C#] public int EndReceive(IAsyncResult asyncResult);  
13 [C++] public: int EndReceive(IAsyncResult* asyncResult);  
14 [VB] Public Function EndReceive(ByVal asyncResult As IAsyncResult) As  
15 Integer  
16 [JScript] public function EndReceive(asyncResult : IAsyncResult) : int;  
17  
18 Description
```

19       Ends a pending asynchronous read.

20       *Return Value:* The number of bytes received. The pending asynchronous read.

21      EndReceiveFrom

```
22  
23 [C#] public int EndReceiveFrom(IAsyncResult asyncResult, ref EndPoint  
24 endPoint);  
25 [C++] public: int EndReceiveFrom(IAsyncResult* asyncResult, EndPoint**
```

```
1  endPoint);  
2  [VB] Public Function EndReceiveFrom(ByVal asyncResult As IAsyncResult,  
3  ByRef endPoint As EndPoint) As Integer  
4  [JScript] public function EndReceiveFrom(asyncResult : IAsyncResult, endPoint :  
5  EndPoint) : int;  
6  
7  Description
```

8       Ends a pending asynchronous read from a specific endpoint.

9       *Return Value*: If successful, the number of bytes received. If unsuccessful, then if  
10      the socket was closed with **System.Net.Sockets.Socket.Close** , returns 0;  
11      otherwise, returns an invalid socket error. The pending asynchronous read. The  
12      source endpoint.

```
13      EndSend
```

```
14  
15  [C#] public int EndSend(IAsyncResult asyncResult);  
16  [C++] public: int EndSend(IAsyncResult* asyncResult);  
17  [VB] Public Function EndSend(ByVal asyncResult As IAsyncResult) As Integer  
18  [JScript] public function EndSend(asyncResult : IAsyncResult) : int;  
19  
20  Description
```

21       Ends a pending asynchronous send.

22       *Return Value*: If successful, the number of bytes sent to the socket; otherwise, an  
23      invalid socket error is returned. The pending asynchronous send.

```
24      EndSendTo
```

```
1  
2 [C#] public int EndSendTo(IAsyncResult asyncResult);  
3 [C++] public: int EndSendTo(IAsyncResult* asyncResult);  
4 [VB] Public Function EndSendTo(ByVal asyncResult As IAsyncResult) As  
5 Integer  
6 [JScript] public function EndSendTo(asyncResult : IAsyncResult) : int;  
7
```

#### 8 *Description*

9     Ends a pending asynchronous send to a specific location.

10     *Return Value*: If successful, the number of bytes sent; otherwise, an invalid socket  
11     error is returned. The pending asynchronous send.

#### 12     Finalize

```
13  
14 [C#] ~Socket();  
15 [C++] ~Socket();  
16 [VB] Overrides Protected Sub Finalize()  
17 [JScript] protected override function Finalize();  
18
```

#### 19 *Description*

20     Frees resources used by the **System.Net.Sockets.Socket** class.

21     The **System.Net.Sockets.Socket** class finalizer calls the  
22 **System.Net.Sockets.Socket.Close** method to close the socket and free resources  
23 associated with the socket.

#### 24     GetHashCode

1  
2 [C#] public override int GetHashCode();  
3 [C++] public: int GetHashCode();  
4 [VB] Overrides Public Function GetHashCode() As Integer  
5 [JScript] public override function GetHashCode() : int;  
6

7 *Description*

8  
9     GetSocketOption

10  
11 [C#] public object GetSocketOption(SocketOptionLevel optionLevel,  
12     SocketOptionName optionName);  
13 [C++] public: Object\* GetSocketOption(SocketOptionLevel optionLevel,  
14     SocketOptionName optionName);  
15 [VB] Public Function GetSocketOption(ByVal optionLevel As  
16     SocketOptionLevel, ByVal optionName As SocketOptionName) As Object  
17 [JScript] public function GetSocketOption(optionLevel : SocketOptionLevel,  
18     optionName : SocketOptionName) : Object; Gets the value of a socket option.  
19

20 *Description*

21     Gets the value of a specified socket option.

22 *Return Value:* The value of the option. When *optionName* is set to  
23 **System.Net.Sockets.SocketOptionName.Linger** the return value is an instance  
24 of **System.Net.Sockets.LingerOption**. When *optionName* is set to  
25 **System.Net.Sockets.SocketOptionName.AddMembership** or

1      **System.Net.Sockets.SocketOptionName.DropMembership** , the return value is  
2      an instance of **System.Net.Sockets.MulticastOption** . When *optionName* is any  
3      other value, the return value is an integer. One of the  
4      **System.Net.Sockets.SocketOptionLevel** values. One of the  
5      **System.Net.Sockets.SocketOptionName** values.

6      **GetSocketOption**

7  
8      [C#] public void GetSocketOption(SocketOptionLevel optionLevel,  
9      SocketOptionName optionName, byte[] optionValue);  
10     [C++] public: void GetSocketOption(SocketOptionLevel optionLevel,  
11     SocketOptionName optionName, unsigned char optionValue \_\_gc[]);  
12     [VB] Public Sub GetSocketOption(ByVal optionLevel As SocketOptionLevel,  
13     ByVal optionName As SocketOptionName, ByVal optionValue() As Byte)  
14     [JScript] public function GetSocketOption(optionLevel : SocketOptionLevel,  
15     optionName : SocketOptionName, optionValue : Byte[]);

16  
17     *Description*

18       Gets the specified socket option setting. One of the  
19       **System.Net.Sockets.SocketOptionLevel** values. One of the  
20       **System.Net.Sockets.SocketOptionName** values. The buffer to receive the option  
21       setting.

22      **GetSocketOption**

23  
24     [C#] public byte[] GetSocketOption(SocketOptionLevel optionLevel,  
25     SocketOptionName optionName, int optionLength);

```
1 [C++] public: unsigned char GetSocketOption(SocketOptionLevel optionLevel,
2 SocketOptionName optionName, int optionLength) __gc[];
3 [VB] Public Function GetSocketOption( ByVal optionLevel As
4 SocketOptionLevel, ByVal optionName As SocketOptionName, ByVal
5 optionLength As Integer) As Byte()
6 [JScript] public function GetSocketOption(optionLevel : SocketOptionLevel,
7 optionName : SocketOptionName, optionLength : int) : Byte[];
8
```

#### 9 *Description*

10 Gets the value of the specified socket option and returns an array.

11 *Return Value:* An array of bytes containing the value of the socket option.

12 The *optionLength* parameter sets the maximum size of the returned byte  
13 array. If the option value requires fewer bytes, the array will contain only that  
14 many bytes. If the option value requires more bytes, a  
15 **System.Net.Sockets.SocketException** will be thrown. One of the  
16 **System.Net.Sockets.SocketOptionLevel** values. One of the  
17 **System.Net.Sockets.SocketOptionName** values. The length, in bytes, of the  
18 expected return value.

#### 19 IOControl

```
20
21 [C#] public int IOControl(int ioControlCode, byte[] optionInValue, byte[]
22 optionOutValue);
23 [C++] public: int IOControl(int ioControlCode, unsigned char optionInValue
24 __gc[], unsigned char optionOutValue __gc[]);
25 [VB] Public Function IOControl( ByVal ioControlCode As Integer, ByVal
```

```
1 optionInValue() As Byte, ByVal optionOutValue() As Byte) As Integer
2 [JScript] public function IOControl(ioControlCode : int, optionInValue : Byte[],
3 optionOutValue : Byte[]) : int;
```

```
5 Description
```

```
6 Sets low-level operating modes for the socket.
```

```
7 Return Value: The number of bytes returned in optionOutValue .
```

```
8 The
```

```
9 System.Net.Sockets.Socket.IOControl(System.Int32,System.Byte[],System.By
10 te[]]) method provides low-level access to the operating system socket underlying
11 the System.Net.Sockets.Socket class instance. For more information about
12 System.Net.Sockets.Socket.IOControl(System.Int32,System.Byte[],System.By
13 te[]]) , see the documentation in MSDN. The control code of the operation to
14 perform. The input data required by the operation. The output data returned by the
15 operation.
```

```
16 Listen
```

```
17
18 [C#] public void Listen(int backlog);
19 [C++] public: void Listen(int backlog);
20 [VB] Public Sub Listen(ByVal backlog As Integer)
21 [JScript] public function Listen(backlog : int);
```

```
23 Description
```

```
24 Places a socket in a listening state.
```

1        If **System.Net.Sockets.Socket.Listen(System.Int32)** is successful,  
2        **System.Net.Sockets.Socket.Connected** is set to **true** . Maximum length of the  
3        queue of pending connections.

4        Poll

5  
6        [C#] public bool Poll(int microSeconds, SelectMode mode);  
7        [C++] public: bool Poll(int microSeconds, SelectMode mode);  
8        [VB] Public Function Poll( ByVal microSeconds As Integer, ByVal mode As  
9        SelectMode) As Boolean  
10        [JScript] public function Poll(microSeconds : int, mode : SelectMode) : Boolean;

11  
12        *Description*

13        Determines the status of the socket.

14        *Return Value:* Mode Return Value **SelectRead ( System.Net.Sockets.SelectMode**  
15        ) **true** if **System.Net.Sockets.Socket.Listen(System.Int32)** has been called and a  
16        connection is pending, **System.Net.Sockets.Socket.Accept** will succeed -or- **true**  
17        if data is available for reading -or- **true** , if connection has been closed, reset, or  
18        terminated; otherwise, returns **false** . The time to wait for a response, in  
19        microseconds. One of the **System.Net.Sockets.SelectMode** values.

20        Receive

21  
22        [C#] public int Receive(byte[] buffer);  
23        [C++] public: int Receive(unsigned char buffer \_\_gc[]);  
24        [VB] Public Function Receive( ByVal buffer() As Byte) As Integer  
25        [JScript] public function Receive(buffer : Byte[]) : int;

1  
2 *Description*

3        Receives data from a connected socket into a specific location of the  
4 receive buffer.

5 *Return Value:* The number of bytes received. Storage location for received data.

6        **Receive**

7  
8 [C#] public int Receive(byte[] buffer, SocketFlags socketFlags);

9 [C++] public: int Receive(unsigned char buffer \_\_gc[], SocketFlags socketFlags);

10 [VB] Public Function Receive( ByVal buffer() As Byte, ByVal socketFlags As  
11        SocketFlags) As Integer

12 [JScript] public function Receive(buffer : Byte[], socketFlags : SocketFlags) : int;

13  
14 *Description*

15        Receives data from a connected socket into a specific location of the  
16 receive buffer.

17 *Return Value:* The number of bytes received. Storage location for received data. A  
18        bitwise combination of the **System.Net.Sockets.SocketFlags** values.

19        **Receive**

20  
21 [C#] public int Receive(byte[] buffer, int size, SocketFlags socketFlags);

22 [C++] public: int Receive(unsigned char buffer \_\_gc[], int size, SocketFlags  
23        socketFlags);

24 [VB] Public Function Receive( ByVal buffer() As Byte, ByVal size As Integer,  
25        ByVal socketFlags As SocketFlags) As Integer

1 [JScript] public function Receive(buffer : Byte[], size : int, socketFlags :  
2 SocketFlags) : int; Receives data from a connected socket.

3

4 *Description*

5 Receives data from a connected socket into a specific location of the  
6 receive buffer.

7 *Return Value*: The number of bytes received. Storage location for received data.

8 The number of bytes to receive. A bitwise combination of the

9 **System.Net.Sockets.SocketFlags** values.

10 **Receive**

11

12 [C#] public int Receive(byte[] buffer, int offset, int size, SocketFlags socketFlags);

13 [C++] public: int Receive(unsigned char buffer \_\_gc[], int offset, int size,  
14 SocketFlags socketFlags);

15 [VB] Public Function Receive(ByVal buffer() As Byte, ByVal offset As Integer,  
16 ByVal size As Integer, ByVal socketFlags As SocketFlags) As Integer

17 [JScript] public function Receive(buffer : Byte[], offset : int, size : int, socketFlags  
18 : SocketFlags) : int;

19

20 *Description*

21 Receives data from a connected socket into a specific location of the  
22 receive buffer.

23 *Return Value*: The number of bytes received. Storage location for received data.

24 The location in *buffer* to store the received data. The number of bytes to receive. A  
25 bitwise combination of the **System.Net.Sockets.SocketFlags** values.

1           ReceiveFrom

2

3        [C#] public int ReceiveFrom(byte[] buffer, refEndPoint remoteEP);

4        [C++] public: int ReceiveFrom(unsigned char buffer \_\_gc[], EndPoint\*\*

5           remoteEP);

6        [VB] Public Function ReceiveFrom(ByVal buffer() As Byte, ByRef remoteEP As

7           EndPoint) As Integer

8        [JScript] public function ReceiveFrom(buffer : Byte[], remoteEP : EndPoint) : int;

9

10       *Description*

11           Receives a datagram into a specific location in the data buffer and stores the

12           endpoint.

13        *Return Value:* The number of bytes received. Storage location for received data.

14        An **System.NetEndPoint**, passed by reference, that represents the remote server.

15           ReceiveFrom

16

17        [C#] public int ReceiveFrom(byte[] buffer, SocketFlags socketFlags, refEndPoint

18           remoteEP);

19        [C++] public: int ReceiveFrom(unsigned char buffer \_\_gc[], SocketFlags

20           socketFlags, EndPoint\*\* remoteEP);

21        [VB] Public Function ReceiveFrom(ByVal buffer() As Byte, ByVal socketFlags

22           As SocketFlags, ByRef remoteEP As EndPoint) As Integer

23        [JScript] public function ReceiveFrom(buffer : Byte[], socketFlags : SocketFlags,

24           remoteEP : EndPoint) : int;

1  
2 *Description*

3        Receives a datagram into a specific location in the data buffer and stores the  
4        endpoint.

5        *Return Value:* The number of bytes received. Storage location for received data. A  
6        bitwise combination of the **System.Net.Sockets.SocketFlags** values. An  
7        **System.NetEndPoint**, passed by reference, that represents the remote server.

8        ReceiveFrom

9  
10      [C#] public int ReceiveFrom(byte[] buffer, int size, SocketFlags socketFlags, ref  
11        EndPoint remoteEP);

12      [C++] public: int ReceiveFrom(unsigned char buffer \_\_gc[], int size, SocketFlags  
13        socketFlags, EndPoint\*\* remoteEP);

14      [VB] Public Function ReceiveFrom(ByVal buffer() As Byte, ByVal size As  
15        Integer, ByVal socketFlags As SocketFlags, ByRef remoteEP As EndPoint) As  
16        Integer

17      [JScript] public function ReceiveFrom(buffer : Byte[], size : int, socketFlags :  
18        SocketFlags, remoteEP : EndPoint) : int;

19  
20 *Description*

21        Receives a datagram into a specific location in the data buffer and stores the  
22        endpoint.

23        *Return Value:* The number of bytes received. Storage location for received data.  
24        The number of bytes to receive. A bitwise combination of the

1    **System.Net.Sockets.SocketFlags** values. An **System.NetEndPoint**, passed by  
2    reference, that represents the remote server.

3    **ReceiveFrom**

4  
5    [C#] public int ReceiveFrom(byte[] buffer, int offset, int size, SocketFlags  
6    socketFlags, ref EndPoint remoteEP);  
7    [C++] public: int ReceiveFrom(unsigned char buffer \_\_gc[], int offset, int size,  
8    SocketFlags socketFlags, EndPoint\*\* remoteEP);  
9    [VB] Public Function ReceiveFrom(ByVal buffer() As Byte, ByVal offset As  
10   Integer, ByVal size As Integer, ByVal socketFlags As SocketFlags, ByRef  
11   remoteEP As EndPoint) As Integer  
12   [JScript] public function ReceiveFrom(buffer : Byte[], offset : int, size : int,  
13   socketFlags : SocketFlags, remoteEP : EndPoint) : int; Receives a datagram and  
14   stores the source endpoint.

15  
16   *Description*

17        Receives a datagram into a specific location in the data buffer and stores the  
18   endpoint.

19   *Return Value:* The number of bytes received. Storage location for received data.

20   The position in *buffer* to store the received data. The number of bytes to receive. A  
21   bitwise combination of the **System.Net.Sockets.SocketFlags** values. An  
22   **System.NetEndPoint**, passed by reference, that represents the remote server.

23   **Select**

24  
25   [C#] public static void Select(IList checkRead, IList checkWrite, IList checkError,

```
1 int microSeconds);  
2 [C++] public: static void Select(IList* checkRead, IList* checkWrite, IList*  
3 checkError, int microSeconds);  
4 [VB] Public Shared Sub Select( ByVal checkRead As IList, ByVal checkWrite As  
5 IList, ByVal checkError As IList, ByVal microSeconds As Integer)  
6 [JScript] public static function Select( checkRead : IList, checkWrite : IList,  
7 checkError : IList, microSeconds : int);  
8
```

#### 9 *Description*

10     Determines the status of a socket.

11     **System.Net.Sockets.Socket.Select(System.Collections.IList, System.Collections.IList, System.Collections.IList, System.Int32)** is a static method that  
12 determines the status of a set of **System.Net.Sockets.Socket** instances. For  
13 example, to determine which of a set of **System.Net.Sockets.Socket** instances has  
14 data ready to be read, place the **System.Net.Sockets.Socket** in an  
15 **System.Collections.IList** instance, and then call  
16 **System.Net.Sockets.Socket.Select(System.Collections.IList, System.Collections.IList, System.Collections.IList, System.Int32)** . After the call to  
17 **System.Net.Sockets.Socket.Select(System.Collections.IList, System.Collections.IList, System.Collections.IList, System.Int32)** , the **System.Collections.IList**  
18 will contain only those **System.Net.Sockets.Socket** instances with data ready to  
19 be read. A list of **System.Net.Sockets.Socket** instances to check for data to read.  
20 A list of **System.Net.Sockets.Socket** instances to check that are available for  
21 writing. A list of **System.Net.Sockets.Socket** instances to check for errors. The  
22 time to wait for a response, in microseconds.  
23

```
1     Send
2
3 [C#] public int Send(byte[] buffer);
4 [C++] public: int Send(unsigned char buffer __gc[]);
5 [VB] Public Function Send( ByVal buffer() As Byte) As Integer
6 [JScript] public function Send(buffer : Byte[]) : int;
7
```

#### 8 *Description*

9 Sends data to a connected socket, starting at the indicated location in the  
10 data.

11 *Return Value:* The number of bytes sent to the socket. The data to be sent.

```
12     Send
13
14 [C#] public int Send(byte[] buffer, SocketFlags socketFlags);
15 [C++] public: int Send(unsigned char buffer __gc[], SocketFlags socketFlags);
16 [VB] Public Function Send( ByVal buffer() As Byte, ByVal socketFlags As
17    SocketFlags) As Integer
18 [JScript] public function Send(buffer : Byte[], socketFlags : SocketFlags) : int;
19
```

#### 20 *Description*

21 Sends data to a connected socket, starting at the indicated location in the  
22 data.

23 *Return Value:* The number of bytes sent to the socket. The data to be sent. A  
24 bitwise combination of the **System.Net.Sockets.SocketFlags** values.

25 Send

```
1
2 [C#] public int Send(byte[] buffer, int size, SocketFlags socketFlags);
3 [C++] public: int Send(unsigned char buffer __gc[], int size, SocketFlags
4 socketFlags);
5 [VB] Public Function Send(ByVal buffer() As Byte, ByVal size As Integer, ByVal
6 socketFlags As SocketFlags) As Integer
7 [JScript] public function Send(buffer : Byte[], size : int, socketFlags :
8 SocketFlags) : int; Sends data to a connected socket.
```

#### 10 *Description*

11       Sends data to a connected socket, starting at the indicated location in the  
12 data.

13 *Return Value*: The number of bytes sent to the socket. The data to be sent. The  
14 number of bytes to send. A bitwise combination of the  
15 **System.Net.Sockets.SocketFlags** values.

#### 16       Send

```
17
18 [C#] public int Send(byte[] buffer, int offset, int size, SocketFlags socketFlags);
19 [C++] public: int Send(unsigned char buffer __gc[], int offset, int size,
20 SocketFlags socketFlags);
21 [VB] Public Function Send(ByVal buffer() As Byte, ByVal offset As Integer,
22 ByVal size As Integer, ByVal socketFlags As SocketFlags) As Integer
23 [JScript] public function Send(buffer : Byte[], offset : int, size : int, socketFlags :
24 SocketFlags) : int;
```

1  
2 *Description*

3       Sends data to a connected socket, starting at the indicated location in the  
4       data.

5       *Return Value*: The number of bytes sent to the socket. The data to be sent. The  
6       position in the data buffer to begin sending data. The number of bytes to send. A  
7       bitwise combination of the **System.Net.Sockets.SocketFlags** values.

8       SendTo

9  
10      [C#] public int SendTo(byte[] buffer,EndPoint remoteEP);  
11      [C++] public: int SendTo(unsigned char buffer \_\_gc[],EndPoint\* remoteEP);  
12      [VB] Public Function SendTo(Val buffer() As Byte, Val remoteEP As  
13           EndPoint) As Integer  
14      [JScript] public function SendTo(buffer : Byte[], remoteEP : EndPoint) : int;

15  
16 *Description*

17       Sends data to a specific endpoint.

18       *Return Value*: The number of bytes sent. The data to be sent. The destination  
19       location for the data.

20       SendTo

21  
22      [C#] public int SendTo(byte[] buffer,SocketFlags socketFlags,EndPoint  
23           remoteEP);  
24      [C++] public: int SendTo(unsigned char buffer \_\_gc[],SocketFlags socketFlags,  
25           EndPoint\* remoteEP);

```
1 [VB] Public Function SendTo(ByVal buffer() As Byte, ByVal socketFlags As
2 SocketFlags, ByVal remoteEP As EndPoint) As Integer
3
4 [JScript] public function SendTo(buffer : Byte[], socketFlags : SocketFlags,
5 remoteEP : EndPoint) : int;
```

#### 6 *Description*

7 Sends data to a specific endpoint.

8 *Return Value:* The number of bytes sent. The data to be sent. A bitwise  
9 combination of the **System.Net.Sockets.SocketFlags** values. The destination  
10 location for the data.

#### 11 SendTo

```
12
13 [C#] public int SendTo(byte[] buffer, int size, SocketFlags socketFlags, EndPoint
14 remoteEP);
```

```
15 [C++] public: int SendTo(unsigned char buffer __gc[], int size, SocketFlags
16 socketFlags, EndPoint* remoteEP);
```

```
17 [VB] Public Function SendTo(ByVal buffer() As Byte, ByVal size As Integer,
18 ByVal socketFlags As SocketFlags, ByVal remoteEP As EndPoint) As Integer
19
20 [JScript] public function SendTo(buffer : Byte[], size : int, socketFlags :
21 SocketFlags, remoteEP : EndPoint) : int;
```

#### 22 *Description*

23 Sends data to a specific endpoint.

24 *Return Value:* The number of bytes sent. The data to be sent. The number of bytes

1 to send. A bitwise combination of the **System.Net.Sockets.SocketFlags** values.

2 The destination location for the data.

3 **SendTo**

5 [C#] public int SendTo(byte[] buffer, int offset, int size, SocketFlags socketFlags,

6 EndPoint remoteEP);

7 [C++] public: int SendTo(unsigned char buffer \_\_gc[], int offset, int size,

8 SocketFlags socketFlags, EndPoint\* remoteEP);

9 [VB] Public Function SendTo(ByVal buffer() As Byte, ByVal offset As Integer,

10 ByVal size As Integer, ByVal socketFlags As SocketFlags, ByVal remoteEP As

11 EndPoint) As Integer

12 [JScript] public function SendTo(buffer : Byte[], offset : int, size : int, socketFlags

13 : SocketFlags, remoteEP : EndPoint) : int; Sends data to a specific endpoint.

15 *Description*

16 Sends data to a specific endpoint, starting at the indicated location in the  
17 data.

18 *Return Value:* The number of bytes sent. The data to be sent. The position in the  
19 data buffer to begin sending data. The number of bytes to send. A bitwise  
20 combination of the **System.Net.Sockets.SocketFlags** values. The destination  
21 location for the data.

22 **SetSocketOption**

24 [C#] public void SetSocketOption(SocketOptionLevel optionLevel,

25 SocketOptionName optionName, byte[] optionValue);

1 [C++] public: void SetSocketOption(SocketOptionLevel optionLevel,  
2 SocketOptionName optionName, unsigned char optionValue \_\_gc[]);  
3 [VB] Public Sub SetSocketOption(ByVal optionLevel As SocketOptionLevel,  
4 ByVal optionName As SocketOptionName, ByVal optionValue() As Byte)  
5 [JScript] public function SetSocketOption(optionLevel : SocketOptionLevel,  
6 optionName : SocketOptionName, optionValue : Byte[]);

7  
8 *Description*

9 Sets the specified option to the specified value. One of the  
10 **System.Net.Sockets.SocketOptionLevel** values. One of the  
11 **System.Net.Sockets.SocketOptionName** values. The value of the option.

12 SetSocketOption

13  
14 [C#] public void SetSocketOption(SocketOptionLevel optionLevel,  
15 SocketOptionName optionName, int optionValue);  
16 [C++] public: void SetSocketOption(SocketOptionLevel optionLevel,  
17 SocketOptionName optionName, int optionValue);  
18 [VB] Public Sub SetSocketOption(ByVal optionLevel As SocketOptionLevel,  
19 ByVal optionName As SocketOptionName, ByVal optionValue As Integer)  
20 [JScript] public function SetSocketOption(optionLevel : SocketOptionLevel,  
21 optionName : SocketOptionName, optionValue : int); Sets a socket option.

22  
23 *Description*

1        Sets the specified option to the specified value. One of the  
2 **System.Net.Sockets.SocketOptionLevel** values. One of the  
3 **System.Net.Sockets.SocketOptionName** values. The value of the option.  
4        SetSocketOption  
5  
6 [C#] public void SetSocketOption(SocketOptionLevel optionLevel,  
7        SocketOptionName optionName, object optionValue);  
8 [C++] public: void SetSocketOption(SocketOptionLevel optionLevel,  
9        SocketOptionName optionName, Object\* optionValue);  
10 [VB] Public Sub SetSocketOption(ByVal optionLevel As SocketOptionLevel,  
11        ByVal optionName As SocketOptionName, ByVal optionValue As Object)  
12 [JScript] public function SetSocketOption(optionLevel : SocketOptionLevel,  
13        optionName : SocketOptionName, optionValue : Object);

14  
15 *Description*

16        Sets the specified option to the specified value. One of the  
17 **System.Net.Sockets.SocketOptionLevel** values. One of the  
18 **System.Net.Sockets.SocketOptionName** values. An instance of  
19 **System.Net.Sockets.LingerOption** or **System.Net.Sockets.MulticastOption**  
20 containing the value of the option.

21        Shutdown  
22  
23 [C#] public void Shutdown(SocketShutdown how);  
24 [C++] public: void Shutdown(SocketShutdown how);  
25 [VB] Public Sub Shutdown(ByVal how As SocketShutdown)

1 [JScript] public function Shutdown(how : SocketShutdown);

3 *Description*

4 Disables sends and receives on a socket.

5 The **System.Net.Sockets.SocketShutdown** class contains the valid values  
6 for *how*. The valid values are: Value Description Send Disable sending on this  
7 socket. The function that will no longer be allowed.

8 IDisposable.Dispose

10 [C#] void IDisposable.Dispose();

11 [C++] void IDisposable::Dispose();

12 [VB] Sub Dispose() Implements IDisposable.Dispose

13 [JScript] function IDisposable.Dispose();

14       SocketException class (System.Net.Sockets)

15       ToString

18 *Description*

19 The exception that is thrown when a socket error occurs.

20 A **System.Net.Sockets.SocketException** is thrown by the  
21 **System.Net.Sockets.Socket** and **System.Net.Dns** classes when an error occurs  
22 with the network.

23       SocketException

24       *Example Syntax:*

25       ToString

```
1  
2 [C#] public SocketException();  
3 [C++] public: SocketException();  
4 [VB] Public Sub New()  
5 [JScript] public function SocketException(); Initializes a new instance of the  
6 System.Net.Sockets.SocketException class.  
7
```

#### 8 *Description*

9     Initializes a new instance of the **System.Net.Sockets.SocketException**  
10  class with the last operating system error code.

11    The **System.Net.Sockets.SocketException.#ctor** constructor sets the  
12  **System.Net.Sockets.SocketException.ErrorCode** property to the last operating  
13  system socket error that occurred. For more information about socket error codes,  
14  see the Windows Socket Version 2 API error code documentation in MSDN.

15    **SocketException**

16    *Example Syntax:*

17    **ToString**

```
18  
19 [C#] public SocketException(int errorCode);  
20 [C++] public: SocketException(int errorCode);  
21 [VB] Public Sub New(ByVal errorCode As Integer)  
22 [JScript] public function SocketException(errorCode : int);  
23
```

#### 24 *Description*

1       Initializes a new instance of the **System.Net.Sockets.SocketException**  
2    class with the specified error code.

3       The **System.Net.Sockets.SocketException.#ctor** constructor sets the  
4    **System.Net.Sockets.SocketException.ErrorCode** property to *errorCode* . The  
5    error code indicating the error that occurred.

6       **SocketException**

7       *Example Syntax:*

8       **ToString**

9  
10      [C#] protected SocketException(SerializationInfo serializationInfo,  
11       StreamingContext streamingContext);  
12      [C++] protected: SocketException(SerializationInfo\* serializationInfo,  
13       StreamingContext streamingContext);  
14      [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal  
15       streamingContext As StreamingContext)  
16      [JScript] protected function SocketException(serializationInfo : SerializationInfo,  
17       streamingContext : StreamingContext);

18  
19       *Description*

20       Initializes a new instance of the **System.Net.Sockets.SocketException**  
21    class from the specified instances of the  
22    **System.Runtime.Serialization.SerializationInfo** and  
23    **System.Runtime.Serialization.StreamingContext** classes.

24       This constructor implements the  
25    **System.Runtime.Serialization.ISerializable** interface for the

1    **System.Net.Sockets.SocketException** class. A  
2    **System.Runtime.Serialization.SerializationInfo** instance containing the  
3    information required to serialize the new **System.Net.Sockets.SocketException**  
4    instance. A **System.Runtime.Serialization.StreamingContext** containing the  
5    source of the serialized stream associated with the new  
6    **System.Net.Sockets.SocketException** instance.

7        ErrorCode  
8        ToString

9  
10      [C#] public override int ErrorCode {get;}  
11      [C++] public: \_\_property virtual int get\_ErrorCode();  
12      [VB] Overrides Public ReadOnly Property ErrorCode As Integer  
13      [JScript] public function get ErrorCode() : int;

14  
15      *Description*

16        Gets the error code associated with this exception.

17        The **System.Net.Sockets.SocketException.ErrorCode** property contains  
18        the error code associated with the error that caused the exception.

19        HelpLink

20        HResult

21        InnerException

22        Message

23        NativeErrorCode

24        Source

25        StackTrace

1        TargetSite  
2        SocketFlags enumeration (System.Net.Sockets)  
3        ToString

4  
5  
6 *Description*  
7        Provides constant values for socket messages.  
8        ToString

9  
10      [C#] public const SocketFlags DontRoute;  
11      [C++] public: const SocketFlags DontRoute;  
12      [VB] Public Const DontRoute As SocketFlags  
13      [JScript] public var DontRoute : SocketFlags;

14  
15 *Description*  
16      Send without using routing tables.  
17      ToString

18  
19      [C#] public const SocketFlags MaxIOVectorLength;  
20      [C++] public: const SocketFlags MaxIOVectorLength;  
21      [VB] Public Const MaxIOVectorLength As SocketFlags  
22      [JScript] public var MaxIOVectorLength : SocketFlags;

23  
24 *Description*  
25

1 Provides a standard value for the number of WSABUF structures used to  
2 send and receive data.

3 **ToString**

4  
5 [C#] public const SocketFlags None;  
6 [C++] public: const SocketFlags None;  
7 [VB] Public Const None As SocketFlags  
8 [JScript] public var None : SocketFlags;

9  
10 *Description*

11 Use no flags for this call.

12 **ToString**

13  
14 [C#] public const SocketFlags OutOfBand;  
15 [C++] public: const SocketFlags OutOfBand;  
16 [VB] Public Const OutOfBand As SocketFlags  
17 [JScript] public var OutOfBand : SocketFlags;

18  
19 *Description*

20 Process out-of-band data.

21 **ToString**

22  
23 [C#] public const SocketFlags Partial;  
24 [C++] public: const SocketFlags Partial;  
25 [VB] Public Const Partial As SocketFlags

1 [JScript] public var Partial : SocketFlags;

3 *Description*

4 Partial send or receive for message.

5 ToString

7 [C#] public const SocketFlags Peek;

8 [C++] public: const SocketFlags Peek;

9 [VB] Public Const Peek As SocketFlags

10 [JScript] public var Peek : SocketFlags;

12 *Description*

13 Peek at incoming message.

14 SocketOptionLevel enumeration (System.Net.Sockets)

15 ToString

18 *Description*

19 Defines socket option levels for the

20 **System.Net.Sockets.Socket.SetSocketOption(System.Net.Sockets.SocketOptio**

21 **nLevel, System.Net.Sockets.SocketOptionName, System.Int32)** and

22 **System.Net.Sockets.Socket.GetSocketOption(System.Net.Sockets.SocketOptio**

23 **nLevel, System.Net.Sockets.SocketOptionName)** methods.

24 The **System.Net.Sockets.SocketOptionLevel** enumeration defines the

25 socket option levels that can be passed to the

1 **System.Net.Sockets.Socket.SetSocketOption(System.Net.Sockets.SocketOptio**  
2 **nLevel, System.Net.Sockets.SocketOptionName, System.Int32)** and  
3 **System.Net.Sockets.Socket.GetSocketOption(System.Net.Sockets.SocketOptio**  
4 **nLevel, System.Net.Sockets.SocketOptionName)** methods.

5 **ToString**

6

7 [C#] public const SocketOptionLevel IP;  
8 [C++] public: const SocketOptionLevel IP;  
9 [VB] Public Const IP As SocketOptionLevel  
10 [JScript] public var IP : SocketOptionLevel;

11

12 *Description*

13     Socket options apply to IP sockets.

14 **ToString**

15

16 [C#] public const SocketOptionLevel Socket;  
17 [C++] public: const SocketOptionLevel Socket;  
18 [VB] Public Const Socket As SocketOptionLevel  
19 [JScript] public var Socket : SocketOptionLevel;

20

21 *Description*

22     Socket options apply to the socket itself.

23 **ToString**

24

25 [C#] public const SocketOptionLevel Tcp;

```
1 [C++] public: const SocketOptionLevel Tcp;  
2 [VB] Public Const Tcp As SocketOptionLevel  
3 [JScript] public var Tcp : SocketOptionLevel;
```

5 *Description*

6     Socket options apply to TCP sockets.

7     ToString

```
8  
9 [C#] public const SocketOptionLevel Udp;  
10 [C++] public: const SocketOptionLevel Udp;  
11 [VB] Public Const Udp As SocketOptionLevel  
12 [JScript] public var Udp : SocketOptionLevel;
```

14 *Description*

15     Socket options apply to UDP sockets.

16     SocketOptionName enumeration (System.Net.Sockets)

17     ToString

20 *Description*

21     Defines socket option names for the **System.Net.Sockets.Socket** class.

22     ToString

```
23  
24 [C#] public const SocketOptionName AcceptConnection;  
25 [C++] public: const SocketOptionName AcceptConnection;
```

1 [VB] Public Const AcceptConnection As SocketOptionName  
2 [JScript] public var AcceptConnection : SocketOptionName;

3  
4 *Description*

5     Socket is listening.

6     ToString

7  
8 [C#] public const SocketOptionName AddMembership;  
9 [C++] public: const SocketOptionName AddMembership;  
10 [VB] Public Const AddMembership As SocketOptionName  
11 [JScript] public var AddMembership : SocketOptionName;

12  
13 *Description*

14     Add an IP group membership.

15     ToString

16  
17 [C#] public const SocketOptionName AddSourceMembership;  
18 [C++] public: const SocketOptionName AddSourceMembership;  
19 [VB] Public Const AddSourceMembership As SocketOptionName  
20 [JScript] public var AddSourceMembership : SocketOptionName;

21  
22 *Description*

23     Join a source group.

24     ToString

```
1
2 [C#] public const SocketOptionName BlockSource;
3 [C++] public: const SocketOptionName BlockSource;
4 [VB] Public Const BlockSource As SocketOptionName
5 [JScript] public var BlockSource : SocketOptionName;
```

7 *Description*

8     Block data from a source.

9     ToString

```
10
11 [C#] public const SocketOptionName Broadcast;
12 [C++] public: const SocketOptionName Broadcast;
13 [VB] Public Const Broadcast As SocketOptionName
14 [JScript] public var Broadcast : SocketOptionName;
```

16 *Description*

17     Permit sending broadcast messages on the socket.

18     ToString

```
19
20 [C#] public const SocketOptionName BsdUrgent;
21 [C++] public: const SocketOptionName BsdUrgent;
22 [VB] Public Const BsdUrgent As SocketOptionName
23 [JScript] public var BsdUrgent : SocketOptionName;
```

25 *Description*

1        Use urgent data as defined in RFC-1222. This option can be set only once,  
2 and once set, cannot be turned off.

3        **ToString**

4

5        [C#] public const SocketOptionName ChecksumCoverage;  
6        [C++] public: const SocketOptionName ChecksumCoverage;  
7        [VB] Public Const ChecksumCoverage As SocketOptionName  
8        [JScript] public var ChecksumCoverage : SocketOptionName;

9

10      *Description*

11      Set or get UDP checksum coverage.

12      **ToString**

13

14      [C#] public const SocketOptionName Debug;  
15      [C++] public: const SocketOptionName Debug;  
16      [VB] Public Const Debug As SocketOptionName  
17      [JScript] public var Debug : SocketOptionName;

18

19      *Description*

20      Record debugging information.

21      **ToString**

22

23      [C#] public const SocketOptionName DontFragment;  
24      [C++] public: const SocketOptionName DontFragment;  
25      [VB] Public Const DontFragment As SocketOptionName

1 [JScript] public var DontFragment : SocketOptionName;

2

3 *Description*

4 Do not fragment IP datagrams.

5 ToString

6

7 [C#] public const SocketOptionName DontLinger;

8 [C++] public: const SocketOptionName DontLinger;

9 [VB] Public Const DontLinger As SocketOptionName

10 [JScript] public var DontLinger : SocketOptionName;

11

12 *Description*

13 Close socket gracefully without lingering.

14 ToString

15

16 [C#] public const SocketOptionName DontRoute;

17 [C++] public: const SocketOptionName DontRoute;

18 [VB] Public Const DontRoute As SocketOptionName

19 [JScript] public var DontRoute : SocketOptionName;

20

21 *Description*

22 Do not route; send directly to interface addresses.

23 ToString

24

25 [C#] public const SocketOptionName DropMembership;

1 [C++] public: const SocketOptionName DropMembership;  
2 [VB] Public Const DropMembership As SocketOptionName  
3 [JScript] public var DropMembership : SocketOptionName;

5 *Description*

6 Drop an IP group membership.

7 *ToString*

8  
9 [C#] public const SocketOptionName DropSourceMembership;  
10 [C++] public: const SocketOptionName DropSourceMembership;  
11 [VB] Public Const DropSourceMembership As SocketOptionName  
12 [JScript] public var DropSourceMembership : SocketOptionName;

14 *Description*

15 Drop a source group.

16 *ToString*

17  
18 [C#] public const SocketOptionName Error;  
19 [C++] public: const SocketOptionName Error;  
20 [VB] Public Const Error As SocketOptionName  
21 [JScript] public var Error : SocketOptionName;

23 *Description*

24 Get error status and clear.

25 *ToString*

```
1  
2 [C#] public const SocketOptionName ExclusiveAddressUse;  
3 [C++] public: const SocketOptionName ExclusiveAddressUse;  
4 [VB] Public Const ExclusiveAddressUse As SocketOptionName  
5 [JScript] public var ExclusiveAddressUse : SocketOptionName;  
6  
7 Description
```

8       Enables a socket to be bound for exclusive access.

```
9       ToString
```

```
10  
11 [C#] public const SocketOptionName Expedited;  
12 [C++] public: const SocketOptionName Expedited;  
13 [VB] Public Const Expedited As SocketOptionName  
14 [JScript] public var Expedited : SocketOptionName;  
15  
16 Description
```

17       Use expedited data as defined in RFC-1222. This option can be set only  
18       once, and once set, cannot be turned off.

```
19       ToString
```

```
20  
21 [C#] public const SocketOptionName HeaderIncluded;  
22 [C++] public: const SocketOptionName HeaderIncluded;  
23 [VB] Public Const HeaderIncluded As SocketOptionName  
24 [JScript] public var HeaderIncluded : SocketOptionName;  
25
```

1     *Description*

2         Indicates application is providing the IP header for outgoing datagrams.

3  
4     *ToString*

5  
6     [C#] public const SocketOptionName IPOptions;

7     [C++] public: const SocketOptionName IPOptions;

8     [VB] Public Const IPOptions As SocketOptionName

9     [JScript] public var IPOptions : SocketOptionName;

10  
11    *Description*

12         Specifies IP options to be inserted into outgoing datagrams.

13     *ToString*

14  
15     [C#] public const SocketOptionName IpTimeToLive;

16     [C++] public: const SocketOptionName IpTimeToLive;

17     [VB] Public Const IpTimeToLive As SocketOptionName

18     [JScript] public var IpTimeToLive : SocketOptionName;

19  
20    *Description*

21         Set the IP header time-to-live field.

22     *ToString*

23  
24     [C#] public const SocketOptionName KeepAlive;

25     [C++] public: const SocketOptionName KeepAlive;

```
1 [VB] Public Const KeepAlive As SocketOptionName  
2 [JScript] public var KeepAlive : SocketOptionName;
```

#### 4 | *Description*

5 | Send keep-alives.

6 || ToString

8 [C#] public const SocketOptionName Linger;

[C++] public: const SocketOptionName Linger;

[VB] Public Const Linger As SocketOptionName

11 [JScript] public var Linger : SocketOptionName;

11 [JScript] public var Linger : SocketOptionName;

### Description

linger on close if unsent data is present.

15 || ToString

[C#] public const SocketOptionName MaxConnections;

[C++] public: const SocketOptionName MaxConnections;

[VB] Public Const MaxConnections As SocketOptionName

[JScript] public var MaxConnections : SocketOptionName;

1100

Maximum queue length that can be specified by

System.Net.Sockets.Socket.Listen(System.Int32)

## ToString

```
1  
2 [C#] public const SocketOptionName MulticastInterface;  
3 [C++] public: const SocketOptionName MulticastInterface;  
4 [VB] Public Const MulticastInterface As SocketOptionName  
5 [JScript] public var MulticastInterface : SocketOptionName;
```

7 *Description*

8     Set the interface for outgoing multicast packets.

9     ToString

```
10  
11 [C#] public const SocketOptionName MulticastLoopback;  
12 [C++] public: const SocketOptionName MulticastLoopback;  
13 [VB] Public Const MulticastLoopback As SocketOptionName  
14 [JScript] public var MulticastLoopback : SocketOptionName;
```

16 *Description*

17     IP multicast loopback.

18     ToString

```
19  
20 [C#] public const SocketOptionName MulticastTimeToLive;  
21 [C++] public: const SocketOptionName MulticastTimeToLive;  
22 [VB] Public Const MulticastTimeToLive As SocketOptionName  
23 [JScript] public var MulticastTimeToLive : SocketOptionName;
```

25 *Description*

1 IP multicast time to live.

2 ToString

3

4 [C#] public const SocketOptionName NoChecksum;

5 [C++] public: const SocketOptionName NoChecksum;

6 [VB] Public Const NoChecksum As SocketOptionName

7 [JScript] public var NoChecksum : SocketOptionName;

8

9 *Description*

10 Send UDP datagrams with checksum set to zero.

11 ToString

12

13 [C#] public const SocketOptionName NoDelay;

14 [C++] public: const SocketOptionName NoDelay;

15 [VB] Public Const NoDelay As SocketOptionName

16 [JScript] public var NoDelay : SocketOptionName;

17

18 *Description*

19 Disables the Nagle algorithm for send coalescing.

20 ToString

21

22 [C#] public const SocketOptionName OutOfBandInline;

23 [C++] public: const SocketOptionName OutOfBandInline;

24 [VB] Public Const OutOfBandInline As SocketOptionName

25 [JScript] public var OutOfBandInline : SocketOptionName;

1       *Description*

2            Receives out-of-band data in the normal data stream.

3        *ToString*

4

5

6 [C#] public const SocketOptionName PacketInformation;

7 [C++] public: const SocketOptionName PacketInformation;

8 [VB] Public Const PacketInformation As SocketOptionName

9 [JScript] public var PacketInformation : SocketOptionName;

10

11       *Description*

12            Return information about received packets.

13        *ToString*

14

15 [C#] public const SocketOptionName ReceiveBuffer;

16 [C++] public: const SocketOptionName ReceiveBuffer;

17 [VB] Public Const ReceiveBuffer As SocketOptionName

18 [JScript] public var ReceiveBuffer : SocketOptionName;

19

20       *Description*

21            Send low water mark.

22        *ToString*

23

24 [C#] public const SocketOptionName ReceiveLowWater;

25 [C++] public: const SocketOptionName ReceiveLowWater;

1 [VB] Public Const ReceiveLowWater As SocketOptionName  
2 [JScript] public var ReceiveLowWater : SocketOptionName;

3  
4 *Description*

5     Receive low water mark.

6     ToString

7  
8 [C#] public const SocketOptionName ReceiveTimeout;

9 [C++] public: const SocketOptionName ReceiveTimeout;

10 [VB] Public Const ReceiveTimeout As SocketOptionName

11 [JScript] public var ReceiveTimeout : SocketOptionName;

12  
13 *Description*

14     Receive time out.

15     ToString

16  
17 [C#] public const SocketOptionName ReuseAddress;

18 [C++] public: const SocketOptionName ReuseAddress;

19 [VB] Public Const ReuseAddress As SocketOptionName

20 [JScript] public var ReuseAddress : SocketOptionName;

21  
22 *Description*

23     Allows the socket to be bound to an address that is already in use.

24     ToString

25

```
1  
2 [C#] public const SocketOptionName SendBuffer;  
3 [C++] public: const SocketOptionName SendBuffer;  
4 [VB] Public Const SendBuffer As SocketOptionName  
5 [JScript] public var SendBuffer : SocketOptionName;  
6  
7 Description
```

Specifies the total per-socket buffer space reserved for sends. This is unrelated to the maximum message size or the size of a TCP window.

ToString

```
11  
12 [C#] public const SocketOptionName SendLowWater;  
13 [C++] public: const SocketOptionName SendLowWater;  
14 [VB] Public Const SendLowWater As SocketOptionName  
15 [JScript] public var SendLowWater : SocketOptionName;  
16  
17 Description
```

Specifies the total per-socket buffer space reserved for receives. This is unrelated to the maximum message size or the size of a TCP window.

ToString

```
21  
22 [C#] public const SocketOptionName SendTimeout;  
23 [C++] public: const SocketOptionName SendTimeout;  
24 [VB] Public Const SendTimeout As SocketOptionName  
25 [JScript] public var SendTimeout : SocketOptionName;
```

1  
2 *Description*

3       Send timeout.

4       ToString

5  
6 [C#] public const SocketOptionName Type;  
7 [C++] public: const SocketOptionName Type;  
8 [VB] Public Const Type As SocketOptionName  
9 [JScript] public var Type : SocketOptionName;

10  
11 *Description*

12       Get socket type.

13       ToString

14  
15 [C#] public const SocketOptionName TypeOfService;  
16 [C++] public: const SocketOptionName TypeOfService;  
17 [VB] Public Const TypeOfService As SocketOptionName  
18 [JScript] public var TypeOfService : SocketOptionName;

19  
20 *Description*

21       Change the IP header type of service field.

22       ToString

23  
24 [C#] public const SocketOptionName UnblockSource;  
25 [C++] public: const SocketOptionName UnblockSource;

1 [VB] Public Const UnblockSource As SocketOptionName  
2 [JScript] public var UnblockSource : SocketOptionName;

3  
4 *Description*

5 Unblock a previously blocked source.

6 ToString

7  
8 [C#] public const SocketOptionName UseLoopback;  
9 [C++] public: const SocketOptionName UseLoopback;  
10 [VB] Public Const UseLoopback As SocketOptionName  
11 [JScript] public var UseLoopback : SocketOptionName;

12  
13 *Description*

14 Bypass hardware when possible.

15 SocketShutdown enumeration (System.Net.Sockets)

16 ToString

17  
18  
19 *Description*

20 Defines constants used by the

21 **System.Net.Sockets.Socket.Shutdown(System.Net.Sockets.SocketShutdown)**  
22 method.

23 The **System.Net.Sockets.SocketShutdown** class is a helper class that  
24 defines the values that can be passed to the

25

1   **System.Net.Sockets.SocketShutdown**(**System.Net.Sockets.SocketShutdown**)  
2   method.

3    ToString

4  
5   [C#] public const SocketShutdown Both;  
6   [C++] public: const SocketShutdown Both;  
7   [VB] Public Const Both As SocketShutdown  
8   [JScript] public var Both : SocketShutdown;

9  
10   *Description*

11       Shuts down a socket for both sending and receiving. This field is constant.

12    ToString

13  
14   [C#] public const SocketShutdown Receive;  
15   [C++] public: const SocketShutdown Receive;  
16   [VB] Public Const Receive As SocketShutdown  
17   [JScript] public var Receive : SocketShutdown;

18  
19   *Description*

20       Shuts down a socket for receiving. This field is constant.

21    ToString

22  
23   [C#] public const SocketShutdown Send;  
24   [C++] public: const SocketShutdown Send;  
25   [VB] Public Const Send As SocketShutdown

1 [JScript] public var Send : SocketShutdown;

2

3 *Description*

4     Shuts down a socket for sending. This field is constant.

5     SocketType enumeration (System.Net.Sockets)

6     ToString

7

8

9 *Description*

10    Specifies the type of socket an instance of the **System.Net.Sockets.Socket**  
11 class represents.

12     ToString

13

14 [C#] public const SocketType Dgram;

15 [C++] public: const SocketType Dgram;

16 [VB] Public Const Dgram As SocketType

17 [JScript] public var Dgram : SocketType;

18

19 *Description*

20

21     ToString

22

23 [C#] public const SocketType Raw;

24 [C++] public: const SocketType Raw;

25 [VB] Public Const Raw As SocketType

1 [JScript] public var Raw : SocketType;

2  
3 *Description*

4  
5     ToString

6  
7 [C#] public const SocketType Rdm;

8 [C++] public: const SocketType Rdm;

9 [VB] Public Const Rdm As SocketType

10 [JScript] public var Rdm : SocketType;

11  
12 *Description*

13  
14     ToString

15  
16 [C#] public const SocketType Seqpacket;

17 [C++] public: const SocketType Seqpacket;

18 [VB] Public Const Seqpacket As SocketType

19 [JScript] public var Seqpacket : SocketType;

20  
21 *Description*

22  
23     ToString

24  
25 [C#] public const SocketType Stream;

```
1 [C++] public: const SocketType Stream;  
2 [VB] Public Const Stream As SocketType  
3 [JScript] public var Stream : SocketType;
```

5 *Description*

7 ToString

```
9 [C#] public const SocketType Unknown;  
10 [C++] public: const SocketType Unknown;  
11 [VB] Public Const Unknown As SocketType  
12 [JScript] public var Unknown : SocketType;
```

14 *Description*

16 TcpClient class (System.Net.Sockets)

17 ToString

20 *Description*

21 Provides client connections for TCP network services.

22 The **System.Net.Sockets.TcpClient** class builds upon the  
23 **System.Net.Sockets.Socket** class to provide TCP services at a higher level of  
24 abstraction. You can establish a **System.Net.Sockets.TcpClient** connection can be  
25 established by one of two ways.

1           TcpClient  
2        *Example Syntax:*  
3        ToString  
4  
5 [C#] public TcpClient();  
6 [C++] public: TcpClient();  
7 [VB] Public Sub New()  
8 [JScript] public function TcpClient();

9  
10       *Description*  
11       Initializes a new instance of the **System.Net.Sockets.TcpClient** class.  
12       The default constructor sets the IP address to **System.Net.IPAddress.Any**  
13 and the port number to 0.

14           TcpClient  
15        *Example Syntax:*  
16        ToString  
17  
18 [C#] public TcpClient(IPEndPoint localEP);  
19 [C++] public: TcpClient(IPEndPoint\* localEP);  
20 [VB] Public Sub New(ByVal localEP As IPEndPoint)  
21 [JScript] public function TcpClient(localEP : IPEndPoint); Initializes a new  
22 instance of the **System.Net.Sockets.TcpClient** class.

23  
24       *Description*  
25

1       Initializes a new instance of the **System.Net.Sockets.TcpClient** class with  
2 the local endpoint.

3       The *localEP* parameter specifies the local endpoint. This constructor  
4 instantiates a socket and binds that socket to the provided local endpoint. The local  
5 endpoint to which you bind the TCP connection.

6       **TcpClient**

7       *Example Syntax:*

8       **ToString**

9  
10      [C#] public TcpClient(string hostname, int port);  
11      [C++] public: TcpClient(String\* hostname, int port);  
12      [VB] Public Sub New(ByVal hostname As String, ByVal port As Integer)  
13      [JScript] public function TcpClient(hostname : String, port : int);

14  
15      *Description*

16       Initializes a new instance of the **System.Net.Sockets.TcpClient** class and  
17 connects to the specified port on the specified host.

18       This constructor accepts two parameters. Name of the remote host to which  
19 you intend to connect. Port number of the remote host to which you intend to  
20 connect.

21       **Active**

22       **ToString**

23  
24      [C#] protected bool Active {get; set;}  
25      [C++] protected: \_\_property bool get\_Active();protected: \_\_property void

```
1  set_Active(bool);  
2  [VB] Protected Property Active As Boolean  
3  [JScript] protected function get Active() : Boolean;protected function set  
4  Active(Boolean);  
5
```

#### 6 *Description*

7       Gets or set a value that indicates whether a connection has been made.

8       Client

9       ToString

```
10  
11 [C#] protected Socket Client {get; set;}  
12 [C++] protected: __property Socket* get_Client();protected: __property void  
13 set_Client(Socket*);  
14 [VB] Protected Property Client As Socket  
15 [JScript] protected function get Client() : Socket;protected function set  
16 Client(Socket);  
17
```

#### 18 *Description*

19       Gets or sets the underlying **System.Net.Sockets.Socket** instance.

20       Use this property to set or get the underlying **System.Net.Sockets.Socket**  
21 instance of **System.Net.Sockets.TcpClient** .

22       LingerState

23       ToString

```
24  
25 [C#] public LingerOption LingerState {get; set;}
```

```
1 [C++] public: __property LingerOption* get_LingerState();public: __property
2 void set_LingerState(LingerOption* );
3 [VB] Public Property LingerState As LingerOption
4 [JScript] public function get LingerState() : LingerOption;public function set
5 LingerState(LingerOption);
6
```

#### 7 *Description*

8 Gets or sets the value of the connection's

9 **System.Net.Sockets.LingerOption** An instance of the

10 **System.Net.Sockets.LingerOption** class.

11 This property sets or gets a **System.Net.Sockets.LingerOption** instance. If  
12 the **System.Net.Sockets.LingerOption.Enabled** property of the  
13 **System.Net.Sockets.LingerOption** instance is *true*, and data remains to be sent,  
14 then **System.Net.Sockets.TcpClient** remains open even after  
15 **System.Net.Sockets.TcpClient.Close** is called. The amount of time the object  
16 remains open is specified by the **System.Net.Sockets.LingerOption.LingerTime**  
17 property of the **System.Net.Sockets.LingerOption** instance. If the Enabled  
18 property of the **System.Net.Sockets.LingerOption** instance is *false*, then  
19 **System.Net.Sockets.TCPClient** will close immediately, even if data remains to  
20 be sent.

21 NoDelay

22 ToString

24 [C#] public bool NoDelay {get; set;}

25 [C++] public: \_\_property bool get\_NoDelay();public: \_\_property void

```
1  set_NoDelay(bool);  
2  [VB] Public Property NoDelay As Boolean  
3  [JScript] public function get NoDelay() : Boolean;public function set  
4  NoDelay(Boolean);  
5  
6  Description
```

7       Gets or sets a value that enables a delay when send or receive buffers are  
8       full.

9       If **System.Net.Sockets.TcpClient.NoDelay** is set to *false* ,  
10      **System.Net.Sockets.TcpClient** delays the sending and receiving of data while  
11      buffers are full. If **System.Net.Sockets.TcpClient.NoDelay** is set to *true* , then  
12      **System.Net.Sockets.TcpClient** continues writing to buffers even if there is no  
13      capacity left.

```
14      ReceiveBufferSize  
15      ToString
```

```
16  
17  [C#] public int ReceiveBufferSize {get; set;}  
18  [C++] public: __property int get_ReceiveBufferSize();public: __property void  
19  set_ReceiveBufferSize(int);  
20  [VB] Public Property ReceiveBufferSize As Integer  
21  [JScript] public function get ReceiveBufferSize() : int;public function set  
22  ReceiveBufferSize(int);  
23
```

24       *Description*

25       Gets or sets the size of the receive buffer.

1        The **System.Net.Sockets.TcpClient.ReceiveBufferSize** property gets or  
2        sets the amount of bytes that can be stored in the receive buffer. After using  
3        **System.Net.Sockets.TcpClient.GetStream** to retrieve the underlying  
4        **System.Net.Sockets.NetworkStream** instance, you will initiate a read by using  
5        one of the **System.Net.Sockets.NetworkStream** read methods. The amount of  
6        data that can be stored from this read, will be determined by the  
7        **System.Net.Sockets.TcpClient.ReceiveBufferSize** property.

8        **ReceiveTimeout**

9        **ToString**

10  
11        [C#] public int ReceiveTimeout {get; set;}  
12        [C++] public: \_\_property int get\_ReceiveTimeout();public: \_\_property void  
13        set\_ReceiveTimeout(int);  
14        [VB] Public Property ReceiveTimeout As Integer  
15        [JScript] public function get ReceiveTimeout() : int;public function set  
16        ReceiveTimeout(int);

17  
18        *Description*

19        Gets or sets the amount of time a **System.Net.Sockets.TcpClient** object  
20        will wait to receive data once a read is initiated.

21        The **System.Net.Sockets.TcpClient.ReceiveTimeout** property determines  
22        the amount of time a **System.Net.Sockets.TcpClient** object will wait to receive  
23        data after a read is initiated. This time is measured in milliseconds.

24        **SendBufferSize**

25        **ToString**

```
1  
2 [C#] public int SendBufferSize {get; set;}  
3 [C++] public: __property int get_SendBufferSize();public: __property void  
4 set_SendBufferSize(int);  
5 [VB] Public Property SendBufferSize As Integer  
6 [JScript] public function get SendBufferSize() : int;public function set  
7 SendBufferSize(int);  
8
```

#### 9 *Description*

10 Gets or sets the size of the send buffer.  
11 The **System.Net.Sockets.TcpClient.SendBufferSize** property gets or sets  
12 the amount of bytes that can be stored in the send buffer. After using  
13 **System.Net.Sockets.TcpClient.GetStream** to retrieve the underlying  
14 **System.Net.Sockets.NetworkStream** instance, you will initiate a send by using  
15 one of the **System.Net.Sockets.NetworkStream** send methods. The amount of  
16 data that can be sent for each send, will be determined by the  
17 **System.Net.Sockets.TcpClient.SendBufferSize** property.

18 **SendTimeout**

19 **ToString**

```
20  
21 [C#] public int SendTimeout {get; set;}  
22 [C++] public: __property int get_SendTimeout();public: __property void  
23 set_SendTimeout(int);  
24 [VB] Public Property SendTimeout As Integer  
25 [JScript] public function get SendTimeout() : int;public function set
```

1 SendTimeout(int);

3 *Description*

4 Gets or sets the amount of time a **System.Net.Sockets.TcpClient** object  
5 will wait to receive confirmation you initiate a send.

6 After you initiate a send, the underlying **System.Net.Sockets.Socket**  
7 instance returns the number of bytes actually sent to the host. The  
8 **System.Net.Sockets.TcpClient.SendTimeout** property determines the amount of  
9 time a **System.Net.Sockets.TcpClient** object will wait before receiving the  
10 number of bytes returned by the **System.Net.Sockets.Socket** class.

11 Close

13 [C#] public void Close();

14 [C++] public: void Close();

15 [VB] Public Sub Close()

16 [JScript] public function Close();

18 *Description*

19 Closes the TCP connection.

20 Closes the TCP connection.

21 Connect

23 [C#] public void Connect(IPEndPoint remoteEP);

24 [C++] public: void Connect(IPEndPoint\* remoteEP);

25 [VB] Public Sub Connect( ByVal remoteEP As IPEndPoint)

1 [JScript] public function Connect(remoteEP : IPEndPoint);

3 *Description*

4     Connects the client to a remote TCP host using the specified remote  
5     network endpoint.

6     **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)**

7     establishes a TCP connection using the specified network endpoint. Before you  
8     call **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)** , you  
9     must first instantiate an **System.Net.IPEndPoint** object using an IP address and  
10    port number. This is your network end-point. After you have instantiated the  
11    **System.Net.Sockets.IPEndPoint** object, you can then call  
12    **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)** and use  
13    the **System.Net.IPEndPoint** object as its argument. The IP endpoint to which you  
14    intend to connect.

15     **Connect**

16  
17 [C#] public void Connect(IPAddress address, int port);

18 [C++] public: void Connect(IPAddress\* address, int port);

19 [VB] Public Sub Connect(ByVal address As IPAddress, ByVal port As Integer)

20 [JScript] public function Connect(address : IPAddress, port : int);

22 *Description*

23     Connects the client to a remote TCP host using the specified IP Address  
24     and port number.

1                   **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)**  
2 establishes a TCP connection using the specified port and IP address. This method  
3 establishes a connection by calling the  
4                   **System.Net.Sockets.Socket.Connect(System.NetEndPoint)** method of the  
5 underlying **System.Net.Sockets.Socket** instance. After the connection is  
6 established, you can use the underlying **System.Net.Sockets.NetworkStream**  
7 instance to send and receive data. Use **System.Net.Sockets.TcpClient.GetStream**  
8 to obtain the underlying **System.Net.Sockets.NetworkStream** instance. IP  
9 address of the host to which you intend to connect. The port number to which you  
10 intend to connect.

11                   **Connect**

12  
13 [C#] public void Connect(string hostname, int port);  
14 [C++] public: void Connect(String\* hostname, int port);  
15 [VB] Public Sub Connect(ByVal hostname As String, ByVal port As Integer)  
16 [JScript] public function Connect(hostname : String, port : int); Connects the client  
17 to a remote TCP host using the specified host name and port number.

18  
19                   **Description**

20                   Connects the client to the specified port on the specified host.

21                   **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)**  
22 establishes a TCP connection to the host device using the specified *port* and  
23 *hostname* specified. This method establishes a connection by calling the  
24                   **System.Net.Sockets.Socket.Connect(System.NetEndPoint)** method of the  
25 underlying **System.Net.Sockets.Socket** instance. After the connection is

1 established, you can use the underlying **System.Net.Sockets.NetworkStream**  
2 instance to send and receive data. Use **System.Net.Sockets.TcpClient.GetStream**  
3 to obtain the underlying **System.Net.Sockets.NetworkStream** instance. Name of  
4 the remote host to which you intend to connect. Port number of the remote host to  
5 which you intend to connect.

6 **Dispose**

7  
8 [C#] protected virtual void Dispose(bool disposing);  
9 [C++] protected: virtual void Dispose(bool disposing);  
10 [VB] Overridable Protected Sub Dispose(ByVal disposing As Boolean)  
11 [JScript] protected function Dispose(disposing : Boolean);

12  
13 *Description*

14       Releases the unmanaged resources used by the  
15 **System.Net.Sockets.TcpClient** and optionally releases the managed resources.

16       This method releases the underlying **System.Net.Sockets.NetworkStream**  
17 and **System.Net.Sockets.Socket** instances. There are no managed resources that  
18 require releasing. If you call

19       **System.Net.Sockets.TcpClient.Dispose(System.Boolean)** and specify *true* as its  
20 parameter, then be sure to suppress the finalizer by calling the  
21 **System.Net.Sockets.TcpClient.SuppressFinalize** method of the garbage collector.  
22 If **true**, releases both managed and unmanaged resources; if **false**, releases only  
23 unmanaged resources.

24       **Finalize**

```
1  
2 [C#] ~TcpClient();  
3 [C++] ~TcpClient();  
4 [VB] Overrides Protected Sub Finalize()  
5 [JScript] protected override function Finalize();  
6
```

#### 7 *Description*

8       Frees resources used by the **System.Net.Sockets.TcpClient** class.

9       The **System.Net.Sockets.TcpClient** class finalizer calls the  
10      **System.Net.Sockets.TcpClient.Close** method to close the TCP connection, shut  
11      down the network stream, and free the socket.

#### 12     **GetStream**

```
13  
14 [C#] public NetworkStream GetStream();  
15 [C++] public: NetworkStream* GetStream();  
16 [VB] Public Function GetStream() As NetworkStream  
17 [JScript] public function GetStream() : NetworkStream;  
18
```

#### 19     *Description*

20       Returns the stream used to send and receive data.

21     *Return Value:* The underlying **System.Net.Sockets.NetworkStream** instance.

22       **System.Net.Sockets.TcpClient.GetStream** returns the underlying  
23      **System.Net.Sockets.NetworkStream** used to send and receive data.

#### 24     **IDisposable.Dispose**

```
1
2 [C#] void IDisposable.Dispose();
3 [C++] void IDisposable::Dispose();
4 [VB] Sub Dispose() Implements IDisposable.Dispose
5 [JScript] function IDisposable.Dispose();
6     TcpListener class (System.Net.Sockets)
7     ToString
```

#### 10 *Description*

11 Listens for connections from TCP network clients.  
12 The **System.Net.Sockets.TcpListener** class builds upon the  
13 **System.Net.Sockets.Socket** class to provide TCP services at a higher level of  
14 abstraction.

15 TcpListener

#### 16 *Example Syntax:*

17 ToString

```
18
19 [C#] public TcpListener(int port);
20 [C++] public: TcpListener(int port);
21 [VB] Public Sub New(ByVal port As Integer)
22 [JScript] public function TcpListener(port : int);
```

#### 24 *Description*

1       Initializes a new instance of the **System.Net.Sockets.TcpListener** that  
2       listens on the specified port.  
3       The *port* parameter specifies the local port number on which you intend to  
4       listen. This constructor instantiates the underlying  
5       **System.Net.Sockets.IPEndPoint** using the specified port number. When you call  
6       **System.Net.Sockets.TcpListener.Start** , **System.Net.Sockets.TcpListener** uses  
7       the default network interface to listen for connections on the specified port  
8       number. The port on which to listen. If this number is 0, a port number will be  
9       selected at random.

10      **TcpListener**

11      *Example Syntax:*

12      **ToString**

13  
14     [C#] public TcpListener(IPEndPoint localEP);  
15     [C++] public: TcpListener(IPEndPoint\* localeP);  
16     [VB] Public Sub New(ByVal localEP As IPEndPoint)  
17     [JScript] public function TcpListener(localEP : IPEndPoint); Initializes a new  
18       instance of the **System.Net.Sockets.TcpListener** class.

19  
20      *Description*

21       Initializes a new instance of the **System.Net.Sockets.TcpListener** class  
22       with the specified local endpoint.

23       The *localEP* parameter specifies the local endpoint. This constructor sets  
24       the underlying **System.Net.Sockets.IPEndPoint** of  
25       **System.Net.Sockets.TcpListener** to the specified local endpoint. If you call

1    **System.Net.Sockets.TcpListener.Start** , **System.Net.Sockets.TcpListener** will  
2    listen for connections using this endpoint. The localend point to bind the listener  
3    connection to.

4    **TcpListener**

5    *Example Syntax:*

6    **ToString**

7

8    [C#] public TcpListener(IPAddress localaddr, int port);

9    [C++] public: TcpListener(IPAddress\* localaddr, int port);

10    [VB] Public Sub New(ByVal localaddr As IPAddress, ByVal port As Integer)

11    [JScript] public function TcpListener(localaddr : IPAddress, port : int);

12

13    *Description*

14    Initializes a new instance of the **System.Net.Sockets.TcpListener** class  
15    that listens to the specified IP address and port.

16    This constructor instantiates the underlying  
17    **System.Net.Sockets.IPEndPoint** of **System.Net.Sockets.TcpListener** using the  
18    specified IP address and port number. If you call

19    **System.Net.Sockets.TcpListener.Start** , **System.Net.Sockets.TcpListener**  
20    listens for connections using the underlying endpoint. The local IP address to  
21    listen to. The port on which to listen. If this is set to 0, a port number will be  
22    selected at random.

23    **Active**

24    **ToString**

```
1  
2 [C#] protected bool Active {get;}  
3 [C++] protected: __property bool get_Active();  
4 [VB] Protected ReadOnly Property Active As Boolean  
5 [JScript] protected function get Active() : Boolean;  
6  
7 Description
```

Gets or Sets a value that indicates whether

**System.Net.Sockets.TcpListener** is actively listening for client connections.

```
10 LocalEndpoint  
11 ToString
```

```
12  
13 [C#] public EndPoint LocalEndpoint {get;}  
14 [C++] public: __property EndPoint* get_LocalEndpoint();  
15 [VB] Public ReadOnly Property LocalEndpoint As EndPoint  
16 [JScript] public function get LocalEndpoint() : EndPoint;
```

```
17  
18 Description
```

Gets the underlying **System.Net.Sockets.EndPoint** of  
**System.Net.Sockets.TcpListener** .

```
21 Server  
22 ToString
```

```
23  
24 [C#] protected Socket Server {get;}  
25 [C++] protected: __property Socket* get_Server();
```

1 [VB] Protected ReadOnly Property Server As Socket

2 [JScript] protected function get Server() : Socket;

3  
4 *Description*

5     Used by the class to get or set the underlying network socket.

6     Use this property to get the underlying **System.Net.Sockets.Socket**

7 instance of **System.Net.Sockets.TcpListener** .

8     AcceptSocket

9  
10 [C#] public Socket AcceptSocket();

11 [C++] public: Socket\* AcceptSocket();

12 [VB] Public Function AcceptSocket() As Socket

13 [JScript] public function AcceptSocket() : Socket;

14  
15 *Description*

16     Accepts a pending connection request.

17     *Return Value:* A **System.Net.Sockets.Socket** instance to handle the incoming  
18 connection request.

19     **System.Net.Sockets.TcpListener.AcceptSocket** returns a  
20 **System.Net.Sockets.Socket** instance used to facilitate communication with the  
21 connecting client. This **System.Net.Sockets.Socket** is initialized with the IP  
22 Address and port number of the connecting client. You can use any of the  
23 **System.Net.Sockets.Socket.Send(System.Byte[],System.Int32,System.Net.Sock**  
24 **ets.SocketFlags)** and  
25 **System.Net.Sockets.Socket.Receive(System.Byte[],System.Int32,System.Net.S**

1       **ockets.SocketFlags)** methods available in the **System.Net.Sockets.Socket** class to  
2       facilitate communication.

3           **AcceptTcpClient**

4  
5       [C#] public TcpClient AcceptTcpClient();  
6       [C++] public: TcpClient\* AcceptTcpClient();  
7       [VB] Public Function AcceptTcpClient() As TcpClient  
8       [JScript] public function AcceptTcpClient() : TcpClient;

9  
10       *Description*

11           Accepts a pending connection request

12       *Return Value:* A **System.Net.Sockets.TcpClient** instance to handle the incoming  
13       connection request.

14           **System.Net.Sockets.TcpListener.AcceptTcpClient** returns  
15       **System.Net.Sockets.TcpClient** instance used to facilitate communication with the  
16       connecting client. Use **System.Net.Sockets.TcpClient.GetStream** to obtain the  
17       underlying **System.Net.Sockets.NetworkStream** instance of  
18       **System.Net.Sockets.TcpClient**. You can use this  
19       **System.Net.Sockets.TcpClient.NetworkStream** to facilitate communication with  
20       the client.

21           **Finalize**

22  
23       [C#] ~TcpListener();  
24       [C++] ~TcpListener();  
25       [VB] Overrides Protected Sub Finalize()

1 [JScript] protected override function Finalize();

3 *Description*

4     Frees resources used by the **System.Net.Sockets.TcpClient** class.

5     The finalizer for the **System.Net.Sockets.TcpListener** class calls the  
6     **System.Net.Sockets.TcpListener.Stop** method to free the underlying  
7     **System.Net.Sockets.Socket** .

8     Pending

10 [C#] public bool Pending();

11 [C++] public: bool Pending();

12 [VB] Public Function Pending() As Boolean

13 [JScript] public function Pending() : Boolean;

15 *Description*

16     Determine if there are pending connection requests.

17     *Return Value:* **true** if connections are pending; otherwise, **false** .

18     Before **System.Net.Sockets.TcpListener** can poll for client connection  
19     attempts, it must first initialize its underlying **System.Net.Sockets.Socket** instance  
20     to listen for incoming connection requests. The way you inititailize the underlying  
21     **System.Net.Sockets.Socket** is by calling the  
22     **System.Net.Sockets.TcpListener.Start** method.

23     Start

25 [C#] public void Start();

1 [C++] public: void Start();

2 [VB] Public Sub Start()

3 [JScript] public function Start();

5 *Description*

6 Starts listening to network requests.

7 **System.Net.Sockets.TcpListener.Start** initializes the underlying

8 **System.Net.Sockets.Socket** instance. After this initialization,

9 **System.Net.Sockets.Start** does the following: Binds the underlying

10 **System.Net.Sockets.Socket** instance using the underlying

11 **System.Net.Sockets.IPEndPoint** instance Begins listening for client connections

12 by calling the **System.Net.Sockets.Socket.Listen(System.Int32)** method of the

13 underlying **System.Net.Sockets.Socket** instance.

14 Stop

16 [C#] public void Stop();

17 [C++] public: void Stop();

18 [VB] Public Sub Stop()

19 [JScript] public function Stop();

21 *Description*

22 Closes the network connection.

23 Close closes the network connection.

24 **UdpClient** class (System.Net.Sockets)

25 **ToString**

1  
2  
3 *Description*

4 Provides UDP (User Datagram Protocol) network services.

5 The **System.Net.Sockets.UdpClient** class builds upon the  
6 **System.Net.Sockets.Socket** class to provide UDP services at a higher level of  
7 abstraction. Because UDP is a connectionless transport protocol, you do not need  
8 to establish a remote host connection prior to sending and receiving data. You do  
9 however, have the option to establish a **System.Net.Sockets.UdpClient**  
10 connection with a remote host using one of the following ways.

11 **UdpClient**

12 *Example Syntax:*

13 **ToString**

14  
15 [C#] public UdpClient();

16 [C++] public: UdpClient();

17 [VB] Public Sub New()

18 [JScript] public function UdpClient(); Initializes a new instance of the

19 **System.Net.Sockets.UdpClient** class.

20  
21 *Description*

22 Initializes a new instance of the **System.Net.Sockets.UdpClient** class.

23 The default constructor sets the IP address to **System.Net.IPEndPoint.Any**  
24 and the port number to 0.

25 **UdpClient**

1       *Example Syntax:*

2        ToString

3

4       [C#] public UdpClient(int port);

5       [C++] public: UdpClient(int port);

6       [VB] Public Sub New(ByVal port As Integer)

7       [JScript] public function UdpClient(port : int);

8

9       *Description*

10       Initializes a new instance of the **System.Net.Sockets.UdpClient** class that  
11       communicates on a specified port.

12       The *port* parameter specifies the local port number on which you intend to  
13       communicate. This constructor instantiates an underlying  
14       **System.Net.Sockets.Socket** and binds that **System.Net.Sockets.Socket** using the  
15       provided port number. This constructor configures the underlying  
16       **System.Net.Sockets.Socket** to use the default local network interface for  
17       communication. The local port number from which you intend to communicate.

18       UdpClient

19       *Example Syntax:*

20       ToString

21

22       [C#] public UdpClient(IPEndPoint localEP);

23       [C++] public: UdpClient(IPEndPoint\* localEP);

24       [VB] Public Sub New(ByVal localEP As IPEndPoint)

25       [JScript] public function UdpClient(localEP : IPEndPoint);

1  
2 *Description*

3       Initializes a new instance of the **System.Net.Sockets.UdpClient** class that  
4       communicates on a specified local endpoint.

5       The *localEP* parameter specifies the local end point. This constructor  
6       instantiates a socket and binds that socket to the local end point provided. The  
7       local endpoint to which you bind the Udp connection.

8       UdpClient

9       *Example Syntax:*

10      ToString

11  
12 [C#] public UdpClient(string hostname, int port);  
13 [C++] public: UdpClient(String\* hostname, int port);  
14 [VB] Public Sub New(ByVal hostname As String, ByVal port As Integer)  
15 [JScript] public function UdpClient(hostname : String, port : int);

16  
17 *Description*

18       Initializes a new instance of the **System.Net.Sockets.UdpClient** class and  
19       connects to a specified remote host on a specified port.

20       This constructor accepts two parameters. The remote host name to which  
21       you intend to connect The remote port number to which you intend to connect

22       Active

23       ToString

24  
25 [C#] protected bool Active {get; set;}

1 [C++] protected: \_\_property bool get\_Active();protected: \_\_property void  
2 set\_Active(bool);

3 [VB] Protected Property Active As Boolean

4 [JScript] protected function get Active() : Boolean;protected function set  
5 Active(Boolean);

6

7 *Description*

8 Gets or sets a value indicating whether a connection to a remote host has  
9 been made.

10 Client

11 ToString

12

13 [C#] protected Socket Client {get; set;}

14 [C++] protected: \_\_property Socket\* get\_Client();protected: \_\_property void  
15 set\_Client(Socket\*);

16 [VB] Protected Property Client As Socket

17 [JScript] protected function get Client() : Socket;protected function set  
18 Client(Socket);

19

20 *Description*

21 Gets or sets the underlying network socket.

22 **System.Net.Sockets.UdpClient** creates a **System.Net.Sockets.Socket**  
23 instance to handle the actual sending and receiving of data over a network.

24 Close

25

```
1  
2 [C#] public void Close();  
3 [C++] public: void Close();  
4 [VB] Public Sub Close()  
5 [JScript] public function Close();  
6
```

7 *Description*

8     Closes the UDP connection.

9     **System.Net.Sockets.TcpClient.Close** disposes the UDP connection.

10    **Connect**

```
11  
12 [C#] public void Connect(IPEndPoint endPoint);  
13 [C++] public: void Connect(IPEndPoint* endPoint);  
14 [VB] Public Sub Connect( ByVal endPoint As IPEndPoint)  
15 [JScript] public function Connect(endPoint : IPEndPoint);  
16
```

17 *Description*

18     Connects the client to a remote UDP host using the specified remote  
19     network endpoint.

20     **System.Net.Sockets.UdpClient.Connect(System.String, System.Int32)**

21     establishes a UDP connection to the host using the specified

22     **System.Net.IPEndPoint** . The network endpoint to which you intend to connect.

23    **Connect**

```
24  
25 [C#] public void Connect(IPAddress addr, int port);
```

```
1 [C++] public: void Connect(IPAddress* addr, int port);  
2 [VB] Public Sub Connect(ByVal addr As IPAddress, ByVal port As Integer)  
3 [JScript] public function Connect(addr : IPAddress, port : int);  
4
```

5 *Description*

6     Connects the client to a remote UDP host using the specified IP Address  
7 and port number.

8     **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)**

9 establishes a UDP connection using the port and IP address specified. This method  
10 establishes a connection by calling the

11     **System.Net.Sockets.Socket.Connect(System.Net.EndPoint)** method of the  
12 underlying **System.Net.Sockets.Socket** instance. After a connection is  
13 established, you can use the

14     **System.Net.Sockets.UdpClient.Send(System.Byte[], System.Int32, System.Net.I  
15 PEndPoint)** and

16     **System.Net.Sockets.UdpClient.Receive(System.Net.IPEndPoint@)** methods to  
17 communicate with the remote host. IP address of the host to which you intend to  
18 connect. The port number to which you intend to connect.

19     **Connect**

```
20  
21 [C#] public void Connect(string hostname, int port);  
22 [C++] public: void Connect(String* hostname, int port);  
23 [VB] Public Sub Connect(ByVal hostname As String, ByVal port As Integer)  
24 [JScript] public function Connect(hostname : String, port : int); Establishes a  
25 connection to a remote host.
```

1            *Description*

2            Establishes a connection to the specified port on the specified remote host.

3            **System.Net.Sockets.TcpClient.Connect(System.String, System.Int32)**

4            establishes a UDP connection to the host machine using the specified *port* and

5            *hostname* . This method establishes a connection by calling the

6            **System.Net.Sockets.Socket.Connect(System.Net.EndPoint)** method of the

7            underlying **System.Net.Sockets.Socket** instance. Once a connection is

8            established, you can use the

9            **System.Net.Sockets.UdpClient.Send(System.Byte[], System.Int32, System.Net.I**

10            **PEndPoint)** and

11            **System.Net.Sockets.UdpClient.Receive(System.Net.IPEndPoint@)** methods to

12            communicate with the remote host. Name of the remote host to which you intend

13            to connect. Port number on the remote host to which you intend to connect.

14            **DropMulticastGroup**

15            [C#] public void DropMulticastGroup(IPAddress multicastAddr);

16            [C++] public: void DropMulticastGroup(IPAddress\* multicastAddr);

17            [VB] Public Sub DropMulticastGroup( ByVal multicastAddr As IPAddress)

18            [JScript] public function DropMulticastGroup(multicastAddr : IPAddress);

19            *Description*

20            Leaves a multicast group.

21            **System.Net.Sockets.UdpClient.DropMulticastGroup(System.Net.IPAD**

22            **dress)** disassociates a **System.Net.Sockets.UdpClient** instance with a multicast

1 group. In order to remove the **System.Net.Sockets.UdpClient** instance from a  
2 multicast group, you must call  
3 **System.Net.Sockets.UdpClient.DropMulticastGroup(System.Net.IPEndPoint)**  
4 using the **System.Net.Sockets.IPEndPoint** of the group from which you intend to  
5 withdrawal. The **System.Net.IPEndPoint** of the multicast group to leave.

6 **JoinMulticastGroup**

7  
8 [C#] public void JoinMulticastGroup(IPAddress multicastAddr);  
9 [C++] public: void JoinMulticastGroup(IPAddress\* multicastAddr);  
10 [VB] Public Sub JoinMulticastGroup( ByVal multicastAddr As IPAddress)  
11 [JScript] public function JoinMulticastGroup(multicastAddr : IPAddress); Adds a  
12 **System.Net.Sockets.UdpClient** to a multicast group.

13  
14 *Description*

15 Adds a **System.Net.Sockets.UdpClient** to a multicast group.

16 **System.Net.Sockets.UdpClient.JoinMulticastGroup(System.Net.IPEndPoint)**  
17 associates the **System.Net.Sockets.UdpClient** instance with a multicast  
18 group. In order to receive data from this group, you must call  
19 **System.Net.Sockets.UdpClient.JoinMulticastGroup(System.Net.IPEndPoint)**  
20 using the **System.Net.Sockets.IPEndPoint** of the group to which you intend to  
21 join. The multicast **System.Net.Sockets.IPEndPoint** of the group you wish to join.

22 **JoinMulticastGroup**

23  
24 [C#] public void JoinMulticastGroup(IPAddress multicastAddr, int timeToLive);  
25 [C++] public: void JoinMulticastGroup(IPAddress\* multicastAddr, int

```
1 timeToLive);  
2 [VB] Public Sub JoinMulticastGroup(ByVal multicastAddr As IPAddress, ByVal  
3 timeToLive As Integer)  
4 [JScript] public function JoinMulticastGroup(multicastAddr : IPAddress,  
5 timeToLive : int);  
6
```

7 *Description*

8 Adds a **System.Net.Sockets.UdpClient** to a multicast group with the  
9 specified Time to Live (TTL).

10 **System.Net.Sockets.UdpClient.JoinMulticastGroup(System.Net.IPAAdd**  
11 **ress)** associates the **System.Net.Sockets.UdpClient** instance with a multicast  
12 group. This overload requires two parameters. The **System.Net.IPAddress** of the  
13 multicast group to join. The Time to Live (TTL)

14 *Receive*

```
15  
16 [C#] public byte[] Receive(ref IPEndPoint remoteEP);  
17 [C++] public: unsigned char Receive(IPEndPoint** remoteEP) __gc[];  
18 [VB] Public Function Receive(ByRef remoteEP As IPEndPoint) As Byte()  
19 [JScript] public function Receive(remoteEP : IPEndPoint) : Byte[];
```

20  
21 *Description*

22 Returns a UDP datagram that was sent by a remote host.

23 *Return Value:* An array of 8-bit unsigned integers that contains the datagram data.

24 An **System.Net.IPEndPoint** representing the remote host from which the data  
25 was sent.

1                   Send

2

3 [C#] public int Send(byte[] dgram, int bytes);

4 [C++] public: int Send(unsigned char dgram \_\_gc[], int bytes);

5 [VB] Public Function Send(ByVal dgram() As Byte, ByVal bytes As Integer) As

6 Integer

7 [JScript] public function Send(dgram : Byte[], bytes : int) : int;

8

9 *Description*

10                  Sends a UDP datagram to a remote host.

11 *Return Value*: The number of bytes sent.

12                  **System.Net.Sockets.UdpClient.Send(System.Byte[],System.Int32, System.**

13 **Net.IPEndPoint)** sends the provided datagram to a remote host. Before using

14 **System.Net.Sockets.UdpClient.Send(System.Byte[],System.Int32, System.Net.I**

15 **PEndPoint)** , you must first use

16 **System.Net.Sockets.UdpClient.Connect(System.String, System.Int32)** to

17 establish a remote host connection. The UDP datagram that you intend to send.

18 The number of bytes in the datagram.

19                  Send

20

21 [C#] public int Send(byte[] dgram, int bytes, IPEndPoint endPoint);

22 [C++] public: int Send(unsigned char dgram \_\_gc[], int bytes, IPEndPoint\*

23 endPoint);

24 [VB] Public Function Send(ByVal dgram() As Byte, ByVal bytes As Integer,

25 ByVal endPoint As IPEndPoint) As Integer

1 [JScript] public function Send(dgram : Byte[], bytes : int, endPoint : IPEndPoint) :  
2 int; Sends a UDP datagram to a remote host.

3

4 *Description*

5 Sends a UDP datagram to the host that is at the remote endpoint.

6 *Return Value*: The number of bytes sent.

7 The

8 **System.Net.Sockets.UdpClient.Send(System.Byte[],System.Int32,System.Net.I**  
9 **PEndPoint)** method establishes a UDP connection to the remote host using the  
10 specified **System.Net.IPEndPoint** , and then sends the datagram. The UDP  
11 datagram to send. The number of bytes in the datagram. An  
12 **System.Net.IPEndPoint** that represents the host and port to which to send the  
13 datagram.

14 Send

15

16 [C#] public int Send(byte[] dgram, int bytes, string hostname, int port);  
17 [C++] public: int Send(unsigned char dgram \_\_gc[], int bytes, String\* hostname,  
18 int port);

19 [VB] Public Function Send(ByVal dgram() As Byte, ByVal bytes As Integer,  
20 ByVal hostname As String, ByVal port As Integer) As Integer

21 [JScript] public function Send(dgram : Byte[], bytes : int, hostname : String, port :  
22 int) : int;

23

24 *Description*

1 Sends a UDP datagram to a specified port on a specified remote host.

2 *Return Value:* The number of bytes sent.

3 **System.Net.Sockets.UdpClient.Send(System.Byte[],System.Int32, System**  
4 **m.Net.IPEndPoint)** establishes a UDP connection to the specified host and port  
5 and then sends the datagram. The data is sent by calling the  
6 **System.Net.Sockets.Socket.SendTo(System.Byte[],System.Int32, System.Int32, System.Net.Sockets.SocketFlags, System.Net.EndPoint)** method of the  
7 underlying **System.Net.Sockets.Socket** instance. The UDP datagram to send.

9

10

11 SYSTEM.SECURITY NAMESPACE

12 The System.Security namespace provides the underlying structure of the  
13 security system, including base classes for permissions. Cryptographic classes in  
14 the namespace are integrated with streams, allowing the classes to be used with  
15 file or network I/O seamlessly. Additionally, security is made available due to  
16 isolated storage that allows code running with restricted permissions to use its own  
17 private file system. Furthermore, a role-based user identity programming model is  
18 unified across different underlying authentication technologies, enabling  
19 applications to write to a single API that works for all the authentication  
20 technologies. As part of this role-based user identity programming model, an  
21 **IIdentity** interface defines the basic functionality of an identity object (which  
22 represents the user on whose behalf the code is running), and an **IPrincipal**  
23 interface defines the basic functionality of a principal object (which represents the  
24 security context of the user on whose behalf the code is running, including that  
25 user's identity (**IIdentity**) and any roles to which the user belongs).

1        The System.Security namespace includes a System.Security.Cryptography  
2        namespace, a System.Security.Cryptography.X509Certificates namespace, a  
3        System.Security.Cryptography.Xml namespace, a System.Security.Permissions  
4        namespace, a System.Security.Policy namespace, and a System.Security.Principal  
5        namespace.

6        The System.Security.Cryptography namespace provides cryptographic  
7        services, including secure encoding and decoding of data, as well as many other  
8        operations, such as hashing, random number generation, and message  
9        authentication. The System.Security.Cryptography.X509Certificates namespace  
10      contains the common language runtime implementation of the Authenticode X.509  
11      v.3 certificate. This certificate is signed with a private key, uniquely and  
12      positively identifying the holder of the certificate. The  
13      System.Security.Cryptography.Xml namespace contains an XML model for use  
14      within the .net framework security system, allowing XML objects to be signed  
15      with a digital signature.

16      The System.Security.Permissions namespace defines classes that control  
17      access to operations and resources based on policy. The System.Security.Policy  
18      namespace contains three types of classes: code groups, membership conditions,  
19      and evidence. These three types of classes are used to create the rules applied by  
20      the common language runtime security policy system. Evidence classes are the  
21      input to security policy and membership conditions are the switches; together  
22      these create policy statements and determine the granted permission set. Policy  
23      levels and code groups are the structure of the policy hierarchy. Code groups are  
24      the encapsulation of a rule and are arranged hierarchically in a policy level. The  
25

1 System.Security.Principal namespace defines a principal object that represents the  
2 security context under which code is running.

3

#### 4 SYSTEM.MANAGEMENT NAMESPACE

5 The System.Management namespace includes numerous objects and  
6 methods that enable rich management through late binding to WMI providers.  
7 Various management functionality and capacity is made available, including  
8 standard querying on management data, events-publication/subscription,  
9 management data retrieval, method invocation (system reboots, process kill, etc.),  
10 system/application configuration, discoverability through common schema,  
11 support for management relationship hierarchies, etc.

12 Numerous objects representing management instrumentation are accessible  
13 through the System.Management namespace (which includes the  
14 System.Management.Instrumentation namespace), from configuration to events  
15 that can be consumed by .NET clients to manage devices. The  
16 System.Management namespace exposes multiple classes (e.g.,  
17 ManagementObject() and ManagementObjectSearcher(), etc.) that allow client  
18 devices to query and enumerate through management classes and invoke methods,  
19 write configurations, subscribe for events, etc. Examples of this functionality  
20 include: SNMP management (traps, etc.); WDM management (windows drivers);  
21 MSI management (windows installer); DS management (directory services);  
22 Security settings management; Network Adapters/bindings management;  
23 Computer System management; Memory, CPU, Processor, and Disks  
24 management; File and directory management; Application management; System  
25 Device management; and so forth.

1

## 2 SYSTEM.SERVICEPROCESS NAMESPACE

3 As discussed above, the System.ServiceProcess namespace contains classes  
4 that allow developers to install and run services. These allow services (e.g.,  
5 Windows NT® operating system services) to be written, installed, and run  
6 relatively easily. These also allow custom installation actions for services to be  
7 packaged into assemblies, easing the installation process.

8

## 9 SYSTEM.RUNTIME.SERIALIZATION NAMESPACE

10 The System.Runtime.Serialization namespace contains classes that can be  
11 used for serializing and deserializing objects in a relatively simple manner.  
12 Serialization is the process of converting an object or a graph of objects into a  
13 linear sequence of bytes for either storage or transmission to another location.  
14 Deserialization is the process of taking in stored information and recreating  
15 objects from it. An ISerializable interface provides a way for classes to control  
16 their own serialization behavior. The System.Runtime.Serialization namespace  
17 includes three additional namespaces: the  
18 System.Runtime.Serialization.Formatters namespace (which contains common  
19 enumerations, interfaces and classes that are used by serialization formatters), the  
20 System.Runtime.Serialization.Formatters.Binary namespace (which contains a  
21 BinaryFormatter class that can be used to serialize and deserialize objects in  
22 binary format), and the System.Runtime.Serialization.Formatters.Soap namespace  
23 (which contains a SoapFormatter class that can be used to serialize and deserialize  
24 objects in the SOAP format).

1           **SYSTEM.MESSAGING NAMESPACE**

2           The System.Messaging namespace provides classes that allow connecting  
3           to message queues on the network, sending messages to queues, receiving  
4           messages from queues, and peeking at (reading without removing) messages from  
5           queues. A MessageQueue class supports both synchronous and asynchronous  
6           retrieval of messages from a queue. Peek and Receive methods on the  
7           MessageQueue class interrupt processing until a message is available in the queue  
8           (or until a specified timeout expires), while BeginPeek and BeginReceive methods  
9           on the MessageQueue class allow processing to continue and use event  
10           notification or callbacks to indicate that a message exists or has arrived in the  
11           queue. MessageQueue use an internal connection cache (or pool), allowing  
12           repeated instantiation of MessageQueue to bind to the same queue very quickly.  
13           Additionally, any object can be sent or received as a message, and the object on  
14           the server and client can be of a different type (as long as the schemas are the  
15           same).

16           The MessageQueue class and Message class are the primary classes that  
17           provide the functionality for message queuing. MessageQueue provides a means  
18           for getting and setting queue properties and default message properties, as well as  
19           initiating message send, peek, or receive operations. MessageQueue also provides  
20           support for transactional message processing. The Message class is used to peek  
21           or receive messages from a queue, or to have fine control over message properties  
22           when sending a message to a queue.

23  
24           **SYSTEM.DIAGNOSTICS NAMESPACE**

1 As discussed above, the System.Diagnostics namespace contains classes  
2 that are used to debug applications and to trace code execution, as well as start  
3 system processes, read and write to event logs, and monitor system performance.  
4 These classes allow, for example, easy publishing of performance counter data and  
5 writing to event logs (e.g., without having to generate separate dll's in order to do  
6 so). A Trace class contains a set of methods and properties that help a developer  
7 trace the execution of code. A Debug class contains a set of methods and  
8 properties that help a developer debug code. A Process class enables access to  
9 local and remote processes, allowing a developer to start and stop local system  
10 processes. An EventLog class enables interaction with Windows® operating  
11 systems event logs, which record information about important software or  
12 hardware events. A PerformanceCounter class represents a Windows NT®  
13 operating system performance counter component that can be used for both  
14 reading existing predefined or custom counters and publishing (writing)  
15 performance data to custom counters.

16  
17 EXEMPLARY COMPUTING SYSTEM AND ENVIRONMENT

18 Fig. 4 illustrates an example of a suitable computing environment 400  
19 within which the programming framework 132 may be implemented (either fully  
20 or partially). The computing environment 400 may be utilized in the computer  
21 and network architectures described herein.

22 The exemplary computing environment 400 is only one example of a  
23 computing environment and is not intended to suggest any limitation as to the  
24 scope of use or functionality of the computer and network architectures. Neither  
25 should the computing environment 400 be interpreted as having any dependency

1 or requirement relating to any one or combination of components illustrated in the  
2 exemplary computing environment 400.

3 The framework 132 may be implemented with numerous other general  
4 purpose or special purpose computing system environments or configurations.  
5 Examples of well known computing systems, environments, and/or configurations  
6 that may be suitable for use include, but are not limited to, personal computers,  
7 server computers, multiprocessor systems, microprocessor-based systems, network  
8 PCs, minicomputers, mainframe computers, distributed computing environments  
9 that include any of the above systems or devices, and so on. Compact or subset  
10 versions of the framework may also be implemented in clients of limited  
11 resources, such as cellular phones, personal digital assistants, handheld computers,  
12 or other communication/computing devices.

13 The framework 132 may be described in the general context of computer-  
14 executable instructions, such as program modules, being executed by one or more  
15 computers or other devices. Generally, program modules include routines,  
16 programs, objects, components, data structures, etc. that perform particular tasks  
17 or implement particular abstract data types. The framework 132 may also be  
18 practiced in distributed computing environments where tasks are performed by  
19 remote processing devices that are linked through a communications network. In  
20 a distributed computing environment, program modules may be located in both  
21 local and remote computer storage media including memory storage devices.

22 The computing environment 400 includes a general-purpose computing  
23 device in the form of a computer 402. The components of computer 402 can  
24 include, by are not limited to, one or more processors or processing units 404, a  
25

1 system memory 406, and a system bus 408 that couples various system  
2 components including the processor 404 to the system memory 406.

3 The system bus 408 represents one or more of several possible types of bus  
4 structures, including a memory bus or memory controller, a peripheral bus, an  
5 accelerated graphics port, and a processor or local bus using any of a variety of  
6 bus architectures. By way of example, such architectures can include an Industry  
7 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an  
8 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)  
9 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a  
10 Mezzanine bus.

11 Computer 402 typically includes a variety of computer readable media.  
12 Such media can be any available media that is accessible by computer 402 and  
13 includes both volatile and non-volatile media, removable and non-removable  
14 media.

15 The system memory 406 includes computer readable media in the form of  
16 volatile memory, such as random access memory (RAM) 410, and/or non-volatile  
17 memory, such as read only memory (ROM) 412. A basic input/output system  
18 (BIOS) 414, containing the basic routines that help to transfer information  
19 between elements within computer 402, such as during start-up, is stored in ROM  
20 412. RAM 410 typically contains data and/or program modules that are  
21 immediately accessible to and/or presently operated on by the processing unit 404.

22 Computer 402 may also include other removable/non-removable,  
23 volatile/non-volatile computer storage media. By way of example, Fig. 4  
24 illustrates a hard disk drive 416 for reading from and writing to a non-removable,  
25 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading

1 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a “floppy  
2 disk”), and an optical disk drive 422 for reading from and/or writing to a  
3 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other  
4 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk  
5 drive 422 are each connected to the system bus 408 by one or more data media  
6 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418,  
7 and optical disk drive 422 can be connected to the system bus 408 by one or more  
8 interfaces (not shown).

9 The disk drives and their associated computer-readable media provide non-  
10 volatile storage of computer readable instructions, data structures, program  
11 modules, and other data for computer 402. Although the example illustrates a  
12 hard disk 416, a removable magnetic disk 420, and a removable optical disk 424,  
13 it is to be appreciated that other types of computer readable media which can store  
14 data that is accessible by a computer, such as magnetic cassettes or other magnetic  
15 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or  
16 other optical storage, random access memories (RAM), read only memories  
17 (ROM), electrically erasable programmable read-only memory (EEPROM), and  
18 the like, can also be utilized to implement the exemplary computing system and  
19 environment.

20 Any number of program modules can be stored on the hard disk 416,  
21 magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by  
22 way of example, an operating system 426, one or more application programs 428,  
23 other program modules 430, and program data 432. Each of the operating system  
24 426, one or more application programs 428, other program modules 430, and  
25

1 program data 432 (or some combination thereof) may include elements of the  
2 programming framework 132.

3 A user can enter commands and information into computer 402 via input  
4 devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse").  
5 Other input devices 438 (not shown specifically) may include a microphone,  
6 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and  
7 other input devices are connected to the processing unit 404 via input/output  
8 interfaces 440 that are coupled to the system bus 408, but may be connected by  
9 other interface and bus structures, such as a parallel port, game port, or a universal  
10 serial bus (USB).

11 A monitor 442 or other type of display device can also be connected to the  
12 system bus 408 via an interface, such as a video adapter 444. In addition to the  
13 monitor 442, other output peripheral devices can include components such as  
14 speakers (not shown) and a printer 446 which can be connected to computer 402  
15 via the input/output interfaces 440.

16 Computer 402 can operate in a networked environment using logical  
17 connections to one or more remote computers, such as a remote computing device  
18 448. By way of example, the remote computing device 448 can be a personal  
19 computer, portable computer, a server, a router, a network computer, a peer device  
20 or other common network node, and so on. The remote computing device 448 is  
21 illustrated as a portable computer that can include many or all of the elements and  
22 features described herein relative to computer 402.

23 Logical connections between computer 402 and the remote computer 448  
24 are depicted as a local area network (LAN) 450 and a general wide area network  
25

1 (WAN) 452. Such networking environments are commonplace in offices,  
2 enterprise-wide computer networks, intranets, and the Internet.

3 When implemented in a LAN networking environment, the computer 402 is  
4 connected to a local network 450 via a network interface or adapter 454. When  
5 implemented in a WAN networking environment, the computer 402 typically  
6 includes a modem 456 or other means for establishing communications over the  
7 wide network 452. The modem 456, which can be internal or external to computer  
8 402, can be connected to the system bus 408 via the input/output interfaces 440 or  
9 other appropriate mechanisms. It is to be appreciated that the illustrated network  
10 connections are exemplary and that other means of establishing communication  
11 link(s) between the computers 402 and 448 can be employed.

12 In a networked environment, such as that illustrated with computing  
13 environment 400, program modules depicted relative to the computer 402, or  
14 portions thereof, may be stored in a remote memory storage device. By way of  
15 example, remote application programs 458 reside on a memory device of remote  
16 computer 448. For purposes of illustration, application programs and other  
17 executable program components such as the operating system are illustrated herein  
18 as discrete blocks, although it is recognized that such programs and components  
19 reside at various times in different storage components of the computing device  
20 402, and are executed by the data processor(s) of the computer.

21 An implementation of the framework 132, and particularly, the API 142 or  
22 calls made to the API 142, may be stored on or transmitted across some form of  
23 computer readable media. Computer readable media can be any available media  
24 that can be accessed by a computer. By way of example, and not limitation,  
25 computer readable media may comprise “computer storage media” and

1 “communications media.” “Computer storage media” include volatile and non-  
2 volatile, removable and non-removable media implemented in any method or  
3 technology for storage of information such as computer readable instructions, data  
4 structures, program modules, or other data. Computer storage media includes, but  
5 is not limited to, RAM, ROM, EEPROM, flash memory or other memory  
6 technology, CD-ROM, digital versatile disks (DVD) or other optical storage,  
7 magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage  
8 devices, or any other medium which can be used to store the desired information  
9 and which can be accessed by a computer.

10 “Communication media” typically embodies computer readable  
11 instructions, data structures, program modules, or other data in a modulated data  
12 signal, such as carrier wave or other transport mechanism. Communication media  
13 also includes any information delivery media. The term “modulated data signal”  
14 means a signal that has one or more of its characteristics set or changed in such a  
15 manner as to encode information in the signal. By way of example, and not  
16 limitation, communication media includes wired media such as a wired network or  
17 direct-wired connection, and wireless media such as acoustic, RF, infrared, and  
18 other wireless media. Combinations of any of the above are also included within  
19 the scope of computer readable media.

20 Alternatively, portions of the framework may be implemented in hardware  
21 or a combination of hardware, software, and/or firmware. For example, one or  
22 more application specific integrated circuits (ASICs) or programmable logic  
23 devices (PLDs) could be designed or programmed to implement one or more  
24 portions of the framework.

1                    **Conclusion**

2                    Although the invention has been described in language specific to structural  
3                    features and/or methodological acts, it is to be understood that the invention  
4                    defined in the appended claims is not necessarily limited to the specific features or  
5                    acts described. Rather, the specific features and acts are disclosed as exemplary  
6                    forms of implementing the claimed invention.